Math2999 final report

Yip Ka Wa (2009137234)

Singular Value Decomposition and its applications

Content

Abstract

Singular value decomposition(SVD) was discovered around 100 years ago and it can be said to be the pinnacle of linear algebra. Nowadays, Singular value decomposition is of great significance, both in mathematics theory and applications. This paper is about the Singular value decomposition and its applications to image compression and latent semantic indexing.

Objective, goals, achievements, future research

The objective of this project is to present and investigate the singular value decomposition and its applications. One of achievement is the successful investigation of SVD'S lower rank approximation. Future research includes why 8×8 block based is generally used in Fourier transform instead of 64×1 column based.

Introduction

This paper first describes the mathematics side of SVD, including the beautiful proof. Then some important results and discovery in image compression will be stated, for example, under what condition when SVD will be useful in image compression.

Comparison of SVD to other methods on image compression like Fourier transform will then be made. Lastly, applications to Latent Semantic Indexing (LSI) and other applications are stated.

Part 1: Mathematics of Singular value decomposition

Suppose we are given a $m\times n$ matrix A, it is not known that whether it has any special properties like diagonal or normal, but we can always decompose it into a product of 3 matrices. This is called the singular value decomposition:

$A=U\Sigma V^*$, where U is $m\times m$ unitary, V is $n\times n$ unitary and Σ is $m\times n$ and only the $(i,i)th$ entries can be nonzero.

The proof of SVD is as beautiful as the theorem itself. Before I proceed to the proof of my claim, it is important to introduce some background mathematics and especially, what the $(i,i)th$ entries of Σ are. The part "some important background mathematics" and the proof of SVD are from the Chap 5 of HKU Math 2303 lecture notes (2009), with slight addition and modification. The Math2303 lecture notes are prepared by Dr. N.K. Tsing.

Some important background mathematics

Positive semidefinite matrix

The main reason why A can be decomposed in such way is because of a very important positive semidefinite matrix, A^*A. It is $n\times n$.

Proof:

$$
\begin{aligned}
&\mathrm{x}^*(A^*A)x \quad \forall \text{nonzero } x\\
&=(Ax)^*(Ax)\\
&=\|Ax\|^2\geq 0\\
&\therefore A^*A \text{ is } n\times n \text{ positive semidefinite matrix}
\end{aligned}
$$

Being positive semidefinite, A^*A has a special property that its eigenvalues are real and nonnegative.

Singular value

As the name of the theorem suggests, the $(i,i)th$ entries of Σ are the singular values.

The definition of singular value is:

Let $A \in F^{m\times n}$. We know the eigenvalues of A^*A, $\lambda_i(A^*A), \geq 0$

The n nonnegative square roots of the eigenvalues of A^*A are called the singular values of A.

$$\sigma_1 = \sqrt{\lambda_1}$$
$$\sigma_2 = \sqrt{\lambda_2}$$
$$\vdots$$
$$\sigma_n = \sqrt{\lambda_n}$$

We shall order them as $\sigma_1(A) \geq \sigma_2(A) \geq ..... \geq \sigma_n(A)(\geq 0)$

$\sigma_1 \geq \sigma_2 \geq ..... \geq \sigma_n(\geq 0)$ singular values of A *iff* $\sigma_1^2 \geq \sigma_2^2 \geq ..... \geq \sigma_n^2$ are eigenvalues of A^*A

Spectrum Decomposition of Hermitian matrix

Suppose $rank\ A = r$ and A^*A is obviously Hermitian. By the spectral decomposition of hermitian matrices (derived from Schur triangularization), we have $A^*A = VDV^*$, where V is unitary and

$D = diag(\sigma_1^2(A), ..., \sigma_n^2(A))$.

Since $rank\ A^*A = rank\ A = r$,

$\Rightarrow rank\ D = r$

$\Rightarrow$ only r of the singular values of A are nonzero.

$\therefore\ \sigma_1(A) \geq ..... \geq \sigma_r(A) > \sigma_{r+1}(A) = ... = \sigma_n(A) = 0$.

With the above mathematics introduced, we can now prove the SVD. The V in SVD is the same as the V in the spectral decomposition of A^*A. The rest of the puzzle is to define what U is.

Singular value decomposition (the statement):

Let $A \in F^{m\times n}$, $\text{rank}A = r$, and $\sigma_1 \geq \cdots \geq \sigma_r$ be the r nonzero singular values.

Then $\text{A} = \text{U}\Sigma\text{V}^*$, where

$A = m\times n$
$U = m\times n \quad unitary$
$V = n\times n \quad unitary$
$\Sigma - m\times n$ matrix,$(i,i)th$ entry is $\sigma_i \ (i = 1,....,r)$, other entries are zero.

Singular value decomposition (the proof):

The proof is consisted of 4 steps.

1. Since A^*A is positive semidefinite, and rank $A^*A = r$, by the spectral decomposition theorem for Hermitian matrices, we have $V^*A^*AV = D$, where $V \in F^{n\times n}$ is unitary and D is the $n\times n$ diagonal matrix $diag(\sigma_1^2(A),......,\sigma_n^2(A))$. Denote the column of V by $v_1,....,v_n$.

2. For $i = r+1,\cdots,n$:

$0 = (i,i)th$ entry of $D = v_i^* A^* A v_i = \langle Av_i, Av_i \rangle = \|Av_i\|_E^2$

$\Rightarrow Av_i = 0$ for $i = r+1,\cdots,n$

Where $\|\square\|_E$ denotes the Euclidean norm.

For $i = 1,\cdots,r$:

3. Let $u_i \in F^m$ by $u_i = \dfrac{1}{\sigma_i} Av_i$. Then for $1 \le i, j \le r$,

$$\left\langle u_j, u_i \right\rangle = \left\langle \frac{1}{\sigma_j} Av_j, \frac{1}{\sigma_i} Av_i \right\rangle = \frac{1}{\sigma_i \sigma_j} v_i^* A^* A v_j$$

$$= \frac{1}{\sigma_i \sigma_j} \times (i, j)\text{th entry of D} = \delta_{ij}$$

Where δ_{ij} is the Kronecker delta. Hence $u_1, \cdots, u_r$ are orthonormal vectors in F^m.

By *Gram-Schmitt Process*, we can extend $u_1, \cdots, u_r$ to orthonormal basis $u_1, \cdots, u_m$ of F^m. We then let $U \in F^{m\times m}$ be the unitary matrix with columns $u_1, \cdots, u_m$.

4. Then the (i, j)th entry of $U\Sigma V^*$ is

$$u_i^* A v_j = \left\langle Av_j, u_i \right\rangle = \begin{cases} \left\langle 0, u_i \right\rangle = 0 & \text{if } j > r \\ \left\langle \sigma_j u_j, u_i \right\rangle = \sigma_j \delta_{ij} & \text{if } j \le r \end{cases}$$

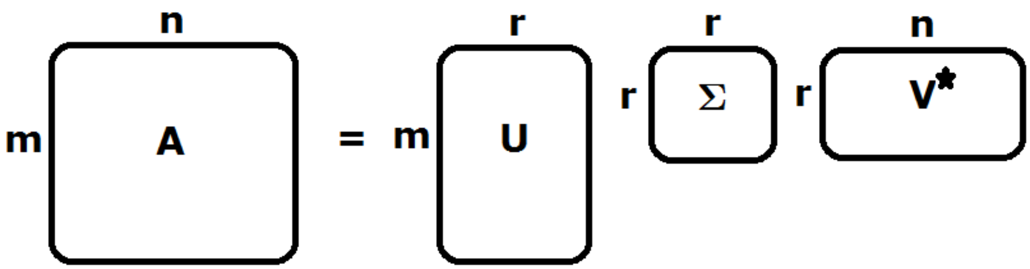
$$\therefore U^* A V = \Sigma \Rightarrow A = U \Sigma V^*$$

Some mathematics results of the SVD

After the proof of the SVD, we can see its many other mathematic properties.

1. Compact form

We can further decomposition A into compact form of singular value decomposition, where Σ becomes square matrix. This form is useful in the application later.



2. Numerical example

Example 1:

Let $A = \begin{pmatrix} 1 & 3 \\ 3 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$

Then computer singular values by:

$$A^*A = \begin{pmatrix} 1 & 3 & 0 & 0 \\ 3 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 3 \\ 3 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 10 & 6 \\ 6 & 10 \end{pmatrix}$$

$\lambda_1\left(A^*A\right)=16 \qquad \lambda_2(A^*A)=4$

$\Rightarrow \sigma_1(A)=4 \qquad \Rightarrow \sigma_2(A)=2$

$$\therefore \Sigma=\begin{pmatrix}4 & 0\\ 0 & 2\\ 0 & 0\\ 0 & 0\end{pmatrix}$$

The columns of V are the eigenvectors of A^*A:

$$V=(v_1\ v_2) \quad v_1=\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix},\ v_2=\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix},\ V=\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}}\end{pmatrix}$$

$U=\left(u_1\ u_2\ u_3\ u_4\right)$

By the definition of u_i (recall the beautiful proof), or you can simply make use of the result $Av_i=\sigma_i u_i$ for i =1,2. I prefer making use of the definition:

$$u_1=\frac{1}{\sigma_1}Av_1=\frac{1}{4}\begin{pmatrix}1 & 3\\ 3 & 1\\ 0 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix}=\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\\ 0\\ 0\end{pmatrix}$$

$$u_2=\frac{1}{\sigma_2}Av_2=\frac{1}{2}\begin{pmatrix}1 & 3\\ 3 & 1\\ 0 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{-1}{\sqrt{2}}\end{pmatrix}=\begin{pmatrix}\frac{-1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\\ 0\\ 0\end{pmatrix}$$

$u_3,u_4\in N(A^*)$

$$A^* = \begin{pmatrix} 1 & 3 & 0 & 0 \\ 3 & 1 & 0 & 0 \end{pmatrix}, \quad \therefore u_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, u_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$U = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \Sigma = \begin{pmatrix} 4 & 0 \\ 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \quad V = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$$

$$\therefore \begin{pmatrix} 1 & 3 \\ 3 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 & 0 \\ 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$$

In compact form:

$$U = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}, \quad V = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$$

$$\therefore \begin{pmatrix} 1 & 3 \\ 3 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$$

Example 2:

$$B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix},\ U = \begin{pmatrix} 0.4046 & 0.9145 \\ 0.9145 & -0.4046 \end{pmatrix},\ \Sigma = \begin{pmatrix} 5.4650 & 0 \\ 0 & 0.3660 \end{pmatrix},$$

$$V = \begin{pmatrix} 0.5760 & -0.8174 \\ 0.8174 & 0.5760 \end{pmatrix}$$

3. Basis of the 4 fundamental subspaces

SVD can be considered to be the pinnacle of linear algebra. Here we will discover how it is related to the fundamental subspaces. Before discussing the connection of SVD with the 4 fundamental subspaces, let us introduce some definition.

4 fundamental subspaces

Readers may have noted that in the first example, I make use of the fact that $u_3, u_4 \in N(A^*)$. In general, a matrix $A \in \mathrm{F}^{m\times n}$ will induce column space $R(A)$, nullspace $N(A)$, row space $R(A^*)$, and the left nullspace $N(A^*)$. They are generally called the 4 fundamental subspaces.

$$A \in \mathrm{F}^{m\times n}$$

$$\begin{aligned} \mathrm{F}^n &= N(A) \oplus N(A)^\perp \\ &= N(A) \oplus R(A^*) \end{aligned}$$

($N(A)^\perp$ is the orthogonal complement of $N(A)$ in F^m)

$$\begin{aligned} \mathrm{F}^m &= R(A) \oplus R(A)^\perp \\ &= R(A) \oplus N(A^*) \end{aligned}$$

F^n is the direct sum of the nullspace and left column space of A.

F^m is the direct sum of the column space and the left nullspace of A.

Right singular vectors, left singular vectors

From SVD, we know that $AV = U\Sigma$. Writing it explicitly,

$$Av_j = \begin{cases} \sigma_j u_j & \text{for } j = 1 \to r; \\ 0 & \text{for } j = r+1 \to n \end{cases}$$

$$A^* u_j = \begin{cases} \sigma_j v_j & \text{for } j = 1 \to r; \\ 0 & \text{for } j = r+1 \to m \end{cases}$$

v_j is the right singular vector and u_j is the left singular vector.

Their connection

$$Av_j = \begin{cases} \sigma_j u_j & \text{for } j = 1 \to r; \\ 0 & \text{for } j = r+1 \to n \end{cases}$$

$$A^* u_j = \begin{cases} \sigma_j v_j & \text{for } j = 1 \to r; \\ 0 & \text{for } j = r+1 \to m \end{cases}$$

$\Rightarrow$

$\{v_1 \ldots\ldots v_n\}$ form an orthonormal basis for F^n

$\{v_{r+1} \ldots\ldots v_n\}$ form an orthonormal basis for $N(A)$

$\{v_1 \ldots\ldots v_r\}$ form an orthonormal basis for $R(A^*)$

$\{u_i \ldots\ldots u_m\}$ form an orthonormal basis for F^m

$\{u_{r+1} \ldots\ldots u_m\}$ form an orthonormal basis for $N(A^*)$

$\{u_1 \ldots\ldots u_r\}$ form an orthonormal basis for $R(A)$

Therefore we can see that the singular vectors form the bases of the 4 fundamental subspaces.

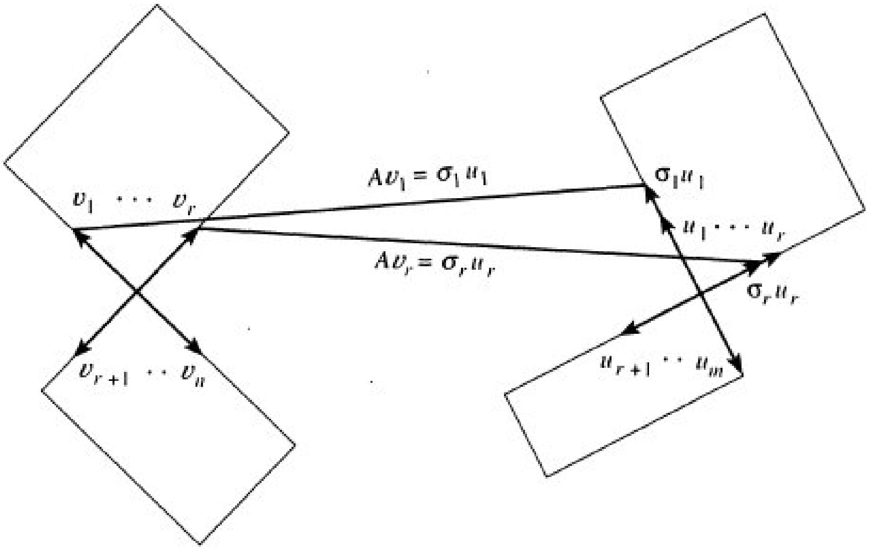


Figure 3. Orthonormal bases that diagonalize A.

(Fundamental subspaces and their basis. Image taken from: Gilbert Strang, the Fundamental theorem of linear algebra, American Mathematical Monthly, Volume 100, Issue 9(Nov.,1993), 848-855)

Part 2: Image compression

After the detailed overview of the mathematics of SVD, here I introduce one of the important applications of SVD, namely image compression. As the image matrix is represented by the real matrix, U,Σ,V are real matrices and we use u^T instead of u^*.

Matrix representation

Digital image is represented by a matrix and the entries are the pixel values. For example in black and white image, 256 represents white color and 0 represents black.

Outer product expansion and truncation of terms

$U\Sigma = (\sigma_1 u_1, \sigma_2 u_2, ..., \sigma_r u_r, ..., \sigma_n u_n)$,where $\sigma_1 \geq ..... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0$

$$\begin{aligned} A = U\Sigma V^T &= \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_n u_n v_n^T \\ &= \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T \end{aligned}$$

In matrix form:

$$A = U\Sigma V^T$$

$$A = \begin{pmatrix} \\ u_1 & u_2 & \cdots & u_m \\ \\ \end{pmatrix} \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ & & \sigma_r \\ 0 & & 0 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{pmatrix},$$

where $\begin{array}{l} \sigma_{r+1}, \cdots, \sigma_n = 0, \text{ if n<m} \\ \sigma_{r+1}, \cdots, \sigma_m = 0, \text{ if m<n} \end{array}$

In compact matrix form:

$$A = U\Sigma V^T$$

$$A = \begin{pmatrix} & & & \\ u_1 & u_2 & \cdots & u_r \\ & & & \end{pmatrix} \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_r \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_r^T \end{pmatrix}$$

$$A = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + \cdots\cdots + u_r\sigma_r v_r^T$$

If r is small enough, the above process will significantly save the memory storage space. But if r is not small enough, the memory used may be increased. The rough condition for memory reduction is:

$$mn \geq r(r + m + n)$$

To ensure that the memory used is reduced, however, we can even do a bit more. We can delete some of the *nonzero* singular values and keep only the first k largest singular values out of the r *nonzero* singular values. r terms in the outer product expansion are reduced to k terms. This original matrix A becomes $\tilde{A}$, which is an approximation matrix of A. This is the meaning of *Truncation of terms*:

$$\tilde{A} = U\tilde{\Sigma}V^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T$$

In matrix form:

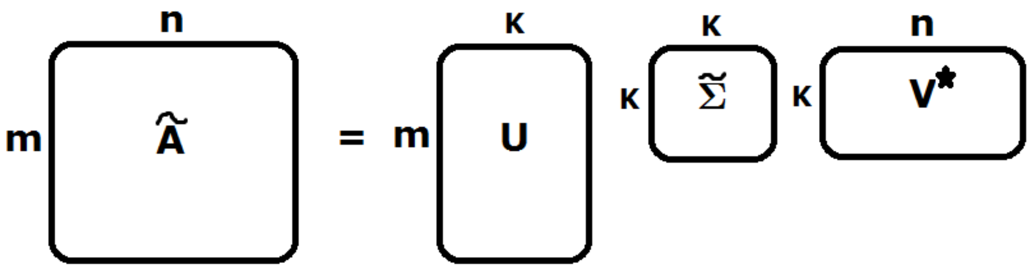
$$\tilde{A} = U\tilde{\Sigma}V^T$$

$$\tilde{A} = \begin{pmatrix} & & & \\ u_1 & u_2 & \cdots & u_m \\ & & & \end{pmatrix} \begin{pmatrix} \sigma_1 & & & 0 \\ & \ddots & & \\ & & \sigma_k & \\ 0 & & & 0 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{pmatrix}$$

In *compact matrix* form:

$$\tilde{A} = \begin{pmatrix} & & & \\ u_1 & u_2 & \cdots & u_k \\ & & & \end{pmatrix} \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_k \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{pmatrix}$$

The memory stored is reduced by considering this storage picture:



$\therefore$ Storage space changes from $mn \to km + k^2 + kn = k(k+m+n)$

If the exact matrix is a square matrix, storage space changes from

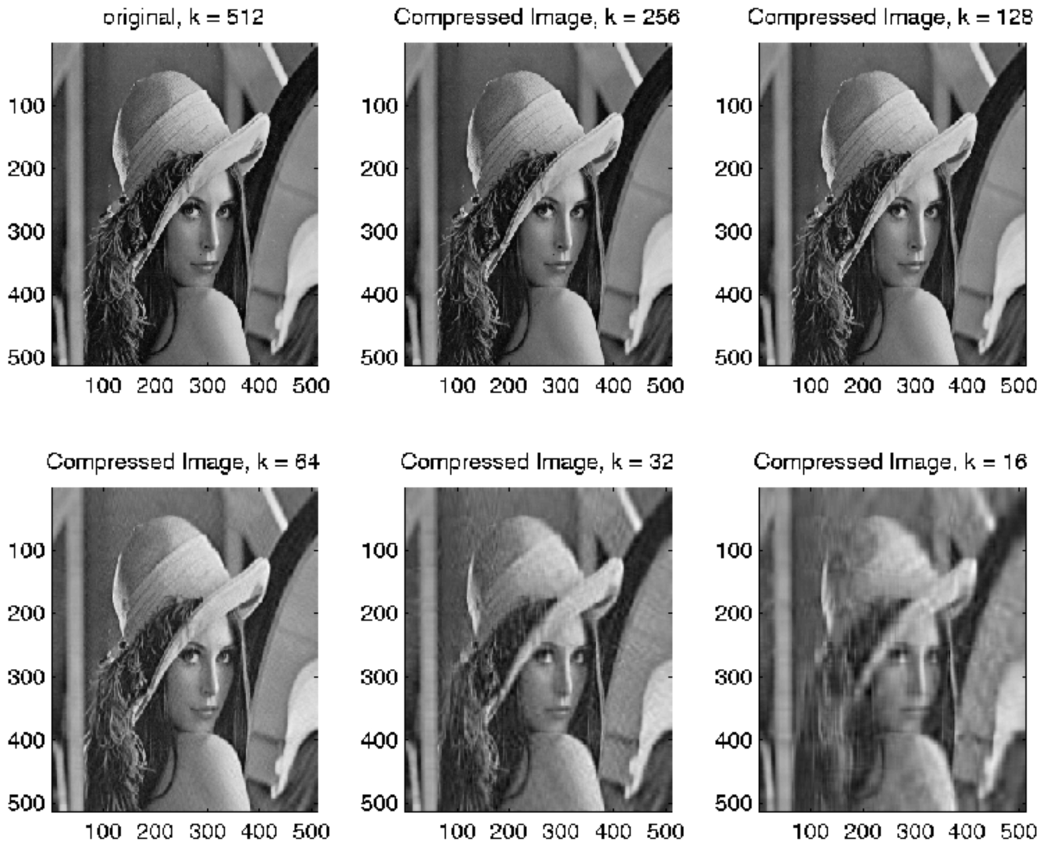
$m^2 \to 2mk + k$

<u>Block-based SVD</u>

Because of different complexity in different region of the image, we usually divide the image matrix into blocks and apply SVD compression to blocks.

<u>Illustrating example of Block-based SVD (James Chen 2000):</u>

Result without block-based

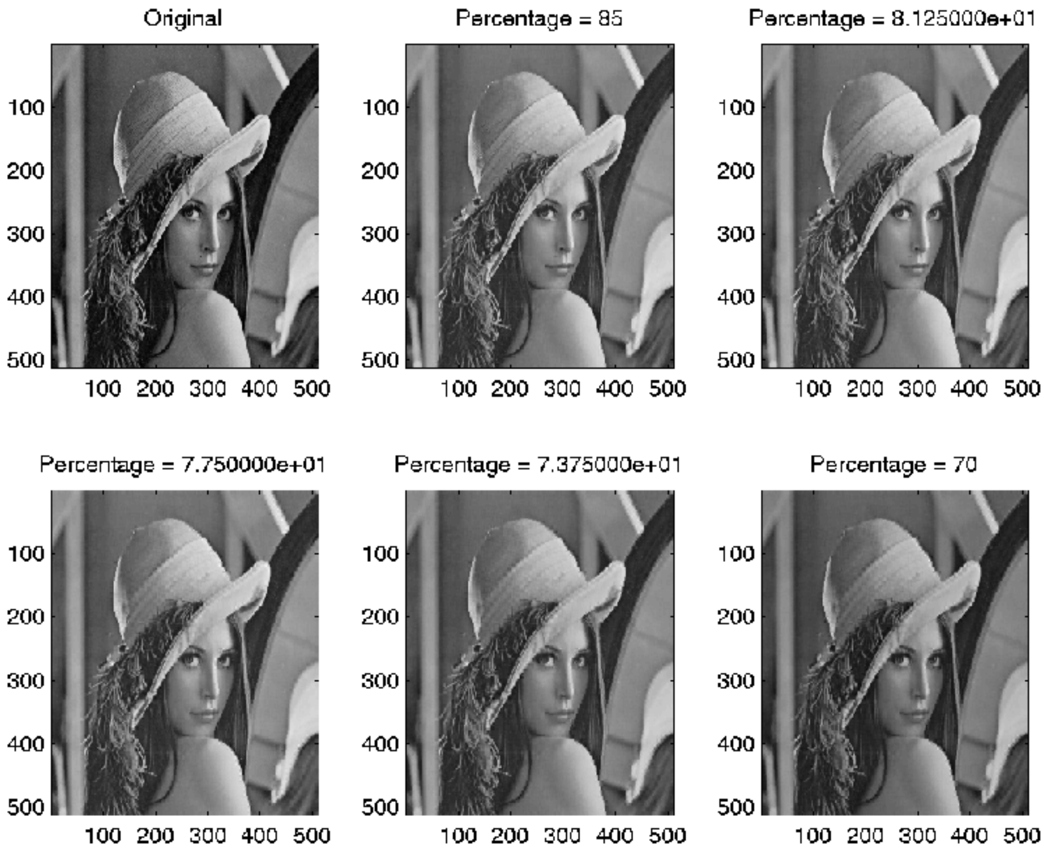


Result with 8×8 blocks SVD

Now divide the matrix into 8×8 blocks and apply SVD into each block. The percentage stated above each image is the percentage of sum of singular values kept over original sum for each block.

i.e. $\dfrac{\sigma_1 + \cdots + \sigma_k}{\sigma_1 + \cdots + \sigma_8}$

The number of singular values kept depends on the singular values distribution of blocks. For example, in some "special" blocks (simple block or see *special topic*), only 1 or 2 singular values are kept for the fixed percentage defined.


Original
Percentage = 85
Percentage = 8.125000e+01
Percentage = 7.750000e+01
Percentage = 7.375000e+01
Percentage = 70
100
200
300
400
500
100 200 300 400 500

Special topic:

Investigation of the kind of images for which low rank SVD approximations can give very good results

By low rank SVD approximation, i mean truncating the last few terms in the outer product expansion of the original matrix. The remaining terms form the approximation matrix. The approximation matrix (image) $\tilde{A}$ turns out to be the best rank k approximation. This is implied by the EY theorem (mentioned later).

Quality of image is mostly measured either by:

1. $\dfrac{\text{Sum of singular values kept}}{\text{original sum of singular values}} = \dfrac{\sigma_1 + \cdots + \sigma_k}{\sigma_1 + \cdots + \sigma_n}$

2. $\dfrac{\text{Frobenius norm of approximated image}}{\text{Frobenius norm of original image}} = \dfrac{\|A_k\|_F}{\|A\|_F}$, where $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$,

is the rank k approximation of A.

Also, the error ratio can then be defined as $\dfrac{\|A - A_k\|_F}{\|A\|_F}$,

3. Error ratio defined by the spectral norm (as in Chen and Duan report).

The error ratio defined by Spectral norm is $\dfrac{\|A - A_k\|_2}{\|A\|_2}$, where $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$,

is the rank k approximation of A.

The general rule of thumb is that for “complex” image the singular values spread out and decay slow, while for “simple” image the first few singular values dominate and singular values decay faster.

However, it is discovered that there are more rules behind. Here I suggest some criteria that the low rank SVD approximations can give good result by listing some examples:

1. The difference (Euclidean norm) between two columns/rows is small compared to the Frobenius norm of the matrix.

2. The difference (Euclidean norm) between columns/rows with zero columns/rows is small compared to the Frobenius norm of the matrix.

3. The columns/rows are near to another columns/row in term of the cosine angle. The cosine angle is near to 1.

In these cases, the Frobenius norm difference between the approximation matrix and the exact matrix is small. From SVD, we know that the best rank $r-1$ approximation to rank r matrix has Frobenius norm difference equal to the $\sqrt{\sigma_r^2}$, it can be said that the r^{th} singular value is small (compared to the Frobenius norm of the original matrix).

<u>*Examples:*</u>

<u>1. The difference (Euclidean norm) between two columns/rows is small compared to the Frobenius norm of the matrix.</u>

$$A=\begin{pmatrix} 99 & 88 & 18 \\ 46 & 77 & 999 \\ 90 & 85 & 20 \end{pmatrix}$$

singular values: $1003.9386, 178.15745, 2.6689437$

It can be observed that the Euclidean norm of difference between row 1 and row 3 is small compared to the Frobenius norm of matrix. The last singular value or even the second one can be truncated without resulting in significant quality loss.

2. The difference (Euclidean norm) between columns/rows with zero columns/rows is small compared to the Frobenius norm of the matrix.

$$B=\begin{pmatrix}10 & 0 & 0\\ 0 & 10 & 0\\ 0 & 0 & 1\end{pmatrix}$$

singular values $=10,10,1$

It can be observed that the Euclidean norm of difference of row 3 and zero row is small compared to the Frobenius norm of matrix. Last singular value can be truncated.

3. The columns/rows are near that another columns/row in term of the cosine angle. The cosine is near 1.

$$C=\begin{pmatrix}1 & 100 & 11\\ 23 & 150 & 15\\ 100 & 200 & 20\end{pmatrix}$$

singular values=284.9111,51.001483,0.7157167

The cosine angle between the 2nd and 3rd columns is close to 1. Last singular value is near 0.

4. (Not obvious example)

$$D=\begin{pmatrix}99 & 108 & 102\\ 36 & 192 & 88\\ 36 & 100 & 57\end{pmatrix}$$

singular values=295.65575,70.040111,0.243387

This example is a not so obvious. The reason behind is

$3\times column3-2\times column1 \, \square \, column2$

5. A 5×5 example:

$$\begin{pmatrix} 1 & 99 & 200 & 89 & 67 \\ 0 & 50 & 100 & 63 & 12 \\ 2 & 90 & 181 & 77 & 13 \\ 2 & 120 & 241 & 11 & 2 \\ 1 & 99 & 199 & 6 & 1 \end{pmatrix}$$

singular values=486.82801, 98.200659, 34.523036, 1.1229673, 0.1487234

Therefore sometimes “complex” image may be well approximated by low-rank matrix.

6. Counter-example

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The columns/rows are near to zero column/row, with Euclidean norm difference being 1. The last singular value is also just 1. However, we have to make the comparison with Frobenius norm of the original matrix. Although the last singular value is small, the Frobenius norm of the original matrix is also small. It turns out that the error ratio (defined above) is large by truncating 1 term. Therefore, when we say that the singular value is small, we should say it in a relative sense.

Theory behind

In the above examples (except Example 5), I assume we keep r-1 singular values out of r singular values, i.e truncating 1 term only. The conclusion can be similarly extended to truncating more terms.

It is implied by the Eckart-Young theorem:

Eckart-Young theorem

Suppose we need to approximate a matrix M with another matrix $\tilde{M}$

having a rank k. In the case that the approximation is based on the minimization of the Frobenius norm of the difference between M and $\tilde{M}$, under the condition that $rank(\tilde{M}) = k$, it turns out that the solution is given by the SVD of M, namely

$\tilde{M} = U\tilde{\Sigma}V^*$, where $\tilde{\Sigma}$ is the same matrix as Σ except that it contains only the k largest singular values (the other singular values are replaced by zero).

(wikipedia: singular value decomposition)

An example illustrating the Eckart-Young theorem:

$$E = \begin{pmatrix} 4 & 2 & 1 \\ 6 & 3 & 2 \\ 2 & 2 & 3 \end{pmatrix}$$

singular values are: 9.0668, 2.1867, 0.10087

Suppose $E = U\Sigma V^*$ (E is rank 3 obviously),

The best rank 2 approximation matrix is given by:

$\tilde{E} = U\tilde{\Sigma}V^*$, as defined in the Eckart-Young theorem.

By computing:

$$\tilde{E} = \begin{pmatrix} 4.0265 & 1.9287687 & 1.0294787 \\ 5.98095 & 3.0510418 & 1.9788769 \\ 2.003438 & 1.9907874 & 3.0038127 \end{pmatrix}$$

(It may not be so obvious that $\tilde{E}$ is a rank 2 matrix)

The computation of $\tilde{E}$ (the best rank 2 approximating matrix) is cumbersome. The $\tilde{E}$ is exactly the low rank SVD approximation matrix in this special topic.

Upper bound of the sum of singular values

However, by observing that the special structure of the matrix E, (column 1 is *nearly* the 2 times of column 2), there is an obvious rank 2 matrix to approximate E, call it $\hat{E}$.

$$\hat{E} = \begin{pmatrix} 4 & 2 & 1 \\ 6 & 3 & 2 \\ 2 & 1 & 3 \end{pmatrix}$$

singular values are: $8.90333, 2.1750195, 0$

It is not a bad approximation because $\left\|E - \hat{E}\right\| = 1$, which is still small compared to the Frobenius norm of E.

The last nonzero singular value of E is $\sigma_3 = 0.10087$

$\therefore \left\|E - \tilde{E}\right\| = 0.10087$,

By Eckart-Young theorem, $\tilde{E}$ is "closest" to E ,so we expect $\left\|E - \tilde{E}\right\| \le \left\|E - \hat{E}\right\|$. This is indeed true as 0.10087<1.

In general, A is a rank r matrix. $\tilde{A}$ is the rank $r-1$ matrix in Eckart young theroem and $\hat{A}$ is the rank $r-1$ matrix we spot.

$\left\|A - \hat{A}\right\| = r \ge \left\|A - \tilde{A}\right\| = \sigma_r$

i.e. $\sigma_r \le r$.

where $\sigma_r (\ge 0)$ is the last nonzero singular value of A.

We can then determine whether a low-rank SVD approximation is good to the image by the upper bound of the last singular value. If the upper bound is small, then we know the last singular value is also small.

Sometimes we can easily spot a matrix $\hat{A}$ of which the rank is 1 lower than A, and observe that the Frobenius norm of the difference matrix between them is small, for example, a column is near to a multiple of another column. As that Frobenius norm is an upper bound for the last nonzero singular value of A, we know the last nonzero singular value of $A$is very small. Therefore it can be truncated to do SVD approximation without much loss of accuracy.

Extension to more term truncations (Upper bound of the sum of singular values)

As can be observed in Example 5, the last two singular values are small.

Actually we can also know the upper bound of the last few singular values sum. Similarly, the above conclusion can be extended to situation when more terms are truncated. What we need to know is whether the sum of the last few singular values are small or not.

For $\hat{A}, \tilde{A}$ being 2 rank lower than that of A, then

$\left\|A-\hat{A}\right\| = s \geq \left\|A-\tilde{A}\right\| = \sqrt{\sigma_{r-1}^2 + \sigma_r^2}$, so when we can spot $\tilde{A}$ being rank 2 lower than A, and we know that the Frobenius norm(s) of the difference matrix is small, then we know that $\sqrt{\sigma_{r-1}^2 + \sigma_r^2}$ is also small. Therefore they can be truncated to do SVD approximation without much loss of accuracy.

For $\hat{A}, \tilde{A}$ being rank k lower than that of A,

$\left\|A-\hat{A}\right\| = t \geq \left\|A-\tilde{A}\right\| = \sqrt{\sigma_{r-k+1}^2 + \cdots + \sigma_{r-1}^2 + \sigma_r^2}$

In conclusion, SVD is a good image compression technique when the original matrix is near to a lower rank matrix.

Part 3: Comparison of different method on image compression

1: Introduction

This report is a comparison between different methods on image compression. In the introduction section, three image compression methods (SVD, Fourier transform, and wavelet transform) will be introduced. After the introduction, the comparison will be made based on the testing results from the Chen and Duan report. Finally, I will draw conclusion on when SVD will be superior in image compression.

1.1 Singular value decomposition

1.1.1 Background theory

The background theory of SVD in image compression is explained in the image compression section before. But to give a better overview of the comparison, the theory part of the technique is inevitable. The following is extracted from the image compression section:

Outer product expansion and truncation of terms

$U\Sigma = (\sigma_1 u_1, \sigma_2 u_2, ..., \sigma_r u_r, ..., \sigma_n u_n)$,where $\sigma_1 \geq ..... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0$

$$\begin{aligned} A = U\Sigma V^T &= \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_n u_n v_n^T \\ &= \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T \end{aligned}$$

In matrix form:

$$A = U\Sigma V^T$$

$$A = \begin{pmatrix} \\ u_1 & u_2 & \cdots & u_m \\ \\ \end{pmatrix} \begin{pmatrix} \sigma_1 & & & 0 \\ & \ddots & & \\ & & \sigma_r & \\ 0 & & & 0 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{pmatrix},$$

where $\begin{matrix} \sigma_{r+1},\cdots,\sigma_n = 0, \text{ if n<m} \\ \sigma_{r+1},\cdots,\sigma_m = 0, \text{ if m<n} \end{matrix}$

In compact matrix form:

$$A = U\Sigma V^T$$

$$A = \begin{pmatrix} \\ u_1 & u_2 & \cdots & u_r \\ \\ \end{pmatrix} \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_r \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_r^T \end{pmatrix}$$

$$A = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + \cdots\cdots + u_r\sigma_r v_r^T$$

If r is small enough, the above process will significantly save the memory storage space. But if r is not small enough, the memory used may be increased. The rough condition for memory reduction is:

$$mn \geq r(r+m+n)$$

To ensure that the memory used is reduced, however, we can even do a bit more. We can delete some of the *nonzero* singular values and keep only the first k largest singular values out of the r *nonzero* singular values. r terms in the outer product expansion are reduced to k terms. This original matrix A becomes $\tilde{A}$, which is an approximation matrix of A. This is the meaning of *Truncation of terms*:

$$\tilde{A} = U\tilde{\Sigma}V^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T$$

In matrix form:

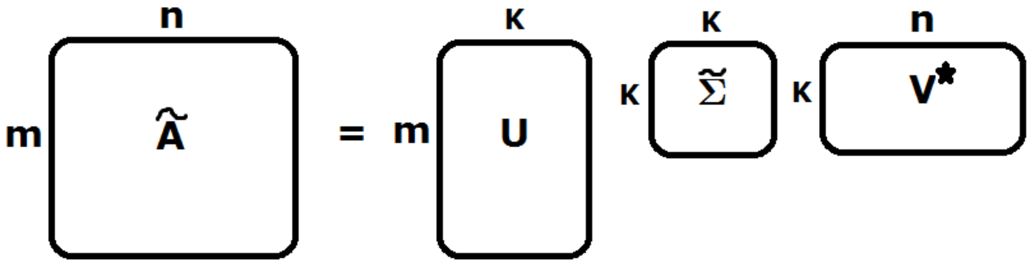
$$\tilde{A} = U\tilde{\Sigma}V^T$$

$$\tilde{A} = \begin{pmatrix} & & & \\ u_1 & u_2 & \cdots & u_m \\ & & & \end{pmatrix} \begin{pmatrix} \sigma_1 & & & 0 \\ & \ddots & & \\ & & \sigma_k & \\ 0 & & & 0 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{pmatrix}$$

In *compact matrix* form:

$$\tilde{A} = \begin{pmatrix} & & & \\ u_1 & u_2 & \cdots & u_k \\ & & & \end{pmatrix} \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_k \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{pmatrix}$$

The memory stored is reduced by considering this storage picture:



$\therefore$ Storage space changes from $mn \rightarrow km + k^2 + kn = k(k + m + n)$

If the exact matrix is a square matrix, storage space changes from

$m^2 \rightarrow 2mk + k$

<u>Block-based SVD</u>

Because of different complexity in different region of the image, we usually divide the image matrix into blocks and apply SVD compression to blocks.

1.2 Fourier transforms

1.2.1 Background theory

JPEG uses Fourier transform. The essence of Fourier transform is the use of Fourier basis and the reduction of coefficients.

Step 1(Lossless step of division into blocks):

Divide 512×512 image into 8×8 block.

Step 2(lossless step of change of basis):

Transform an 8×8 block into a 64×1 vector and change of basis to fourier basis.

$$\begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ x_{64} \end{pmatrix} = c_1 f_1 + c_2 f_2 + \cdots + \cdots + c_{64} f_{64}$$, where f_i is the fourier basis.

$c_1 \geq c_2 \geq \cdots \geq c_{64}$

Some of the Fourier basis are $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ \vdots \end{pmatrix}$ and $\begin{pmatrix} 1 \\ w \\ w^2 \\ w^3 \\ w^4 \\ \vdots \end{pmatrix}$ where $w^n = 1$, w=$e^{2\pi i/n}$, in this case n=64 (Gilbert Strang, 2005).

In general the fourier basis is:

$$\begin{pmatrix}1\\ w\\ w^2\\ w^3\\ w^4\\ \vdots\\ w^{n-1}\end{pmatrix},\begin{pmatrix}1\\ w^2\\ w^4\\ w^6\\ w^8\\ \vdots\\ w^{2(n-1)}\end{pmatrix},\begin{pmatrix}1\\ w^3\\ w^6\\ w^9\\ w^{12}\\ \vdots\\ w^{3(n-1)}\end{pmatrix},\ldots\ldots,\text{ where } w^n=1,\ \text{w=e}^{2\pi i/n},\ \text{n=positive integer}$$

Step 3(lossy process in the reduction of coefficient):

Reduce the number of coefficients. The smaller coefficients are deleted (Gilbert Strang, 2005). For example, an 8×8 matrix block (64 coefficients) can be reduced to using only 20 coefficients.

$$\begin{pmatrix}x_1\\ \vdots\\ \vdots\\ \vdots\\ x_{64}\end{pmatrix}=c_1f_1+c_2f_2+\cdots+c_{21}f_{21}+\cdots+c_{64}f_{64}$$

$$\Rightarrow\begin{pmatrix}x_1'\\ \vdots\\ \vdots\\ \vdots\\ x_{64}'\end{pmatrix}=c_1f_1+\cdots+c_{20}f_{20}$$

In the extreme case, we can use only 1 coefficient to represent a single-color image. The amount of entries stored is significantly reduced.

1.2.2(Types of Fourier transform in image compression)

The Fourier transform in image processing is discrete Fourier transform (DFT) because the images are digital and signals are discrete(Chen and Duan p.12). Discrete Fourier transform (DFT) uses sine and cosine waves while DCT uses cosine waves. In Chen and Duan report, Fast Fourier transform (FFT) which is the image compression scheme found on DCT, is used (Chen and Duan). Also, an image matrix is divided into 8×8 blocks and Fourier transform is applied to the each block.

1.3 Wavelet Transform

1.3.1 Background theory

JPEG 2000 uses wavelet transform. The steps involved are similar while wavelet transform uses wavelet basis. A wavelet is a waveform oscillation with amplitude that starts out at zero, increases, and then decreases back to zero (wikipedia: wavelet). A wavelet can be considered as an advanced form of Fourier transform. Mathematically, an orthonormal wavelet is that can be used to describe a Hilbert basis (wikipedia: wavelet transform) and square integrable function can be represented as wavelet series. Same as Fourier transform, wavelet transform produces as many coefficients as there are pixels in the image (wikipedia: wavelet transform).

For 8 dimensional space, here is some wavelet basis:

$$\begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix},\begin{pmatrix}1\\1\\1\\1\\-1\\-1\\-1\\-1\end{pmatrix},\begin{pmatrix}1\\1\\-1\\-1\\0\\0\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\0\\0\\-1\\-1\\-1\\-1\end{pmatrix},\begin{pmatrix}1\\-1\\0\\0\\0\\0\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\1\\-1\\0\\0\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\0\\0\\1\\-1\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\0\\0\\0\\0\\1\\-1\end{pmatrix}$$

(Gilbert Strang, 2005)

1.3.2 (Types of Wavelet transform in image compression)

There are various types of wavelet transform. For example, continuous wavelet transform, discrete wavelet transform, multiresolution discrete wavelet transform, haar wavelet, Daubechies wavelets. In Chen and Duan report, haar wavelet and daubechies wavelets are used for image compression testing. Note that daubechies wavelets are represented as dbN, where N is the order, and db1 is the haar wavelet. While the mathematics of these wavelets are difficult, in qualitative sense, N is a measure of the accuracy of wavelet(Chen and Duan p.28) and the use of higher N wavelet means better image compression result.

2: Comparison of different methods

2.1 Comparison from general theory (without testing result)

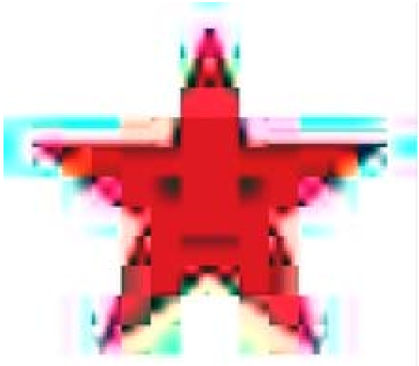
2.1.1 SVD

Advantages

1. Compression speed in SVD is also high.
2. Small change in the input results in small change in the singular matrix so it is more stable (Chen and Duan P.34).
3. Unlike the fourier transform that uses fixed 8×8 block based method, SVD allows the use of other sizes of block so we can decide the optimal size of block to be used. Therefore, it allows us to perform

optimization on memory reduction.

4. Unlike fourier transform and wavelet transform, SVD does not lead to the ringing artifacts. The ringing artifacts appear as the spurious signals near sharp transitions in a signal. In image, bands or "ghosts" appear near edges of objects (wikipedia: ringing artifacts).



(Rings around edges. Picture from: wikipedia: ringing artifacts)

Disadvantages

1. U and V will have to stored together with the singular values (James Chen 2000).

2.1.2 Fourier transform

Advantages

1. Discrete cosine transform(DCT) is 8×8 block-based. Using 8×8 allows fourier basis to be efficiently used. Using bigger blocks such as 64×64 creates huge blocking artifacts and using small block 4×4 reduces the possibility of data compression (stackexchange 2011). Also, 8 is the power of 2 so the transform is quicker to perform (stackexchange 2011). Lastly, 8×8 grouping was based on the maximum size that integrated circuit technology could handle at the time the standard was developed

(steven.w.smith 1998).

According to an email reply by David Austin, the reason why we use 8×8 instead of 64×1 is that the 8x8 blocks have the advantage that the Y,Cr, Cb values of individual pixels are typically closer to constant since the distance between any two pixels is as small as possible. With 64x1, the values at individual pixels could change a lot, and this would mean that we would need to keep more terms in the Fourier transform.

2. The calculation of coefficient is easy:

$c_i = \langle f_i | \psi \rangle = f_i^T \psi$, where f_i is the fourier basis and ψ is the 64×1 vector.

Disadvantages

1. 8×8 block-based method still leads to “observable” blocking artifacts. The blocking artifacts can be observed in the discontinuities at the boundaries of the 8 by 8 blocks (David Austin 2012). For example, the color of a pixel on the edge of a block is influenced by that of any pixel in the block, but not by the adjacent pixel in another block (David Austin 2012).

(Blocking artifacts where 8 by 8 blocks are visible. Picture from:

http://goldfishforthought.blogspot.com/2010/10/not-again.html)

(Blocking artifacts. Picture from: wikipedia: compression artifacts)

2.1.3 Wavelet transform

Advantages

1. Wavelet is superior in most situations in quality.
2. Wavelet transform has the advantage over fourier transform in that it represents both frequency and time. Fourier Transform only shows

which frequencies are used but not when the frequencies occur (Chen and Duan P.17).

3. Wavelet basis is of variable length, and does not block the input image. This property leads to higher compression ratio while avoiding blocking artifacts (Chen and Duan P.29).

Disadvantages

1. Still has ringing artifacts

2.2 Comparison from Chen and Duan testing result (error ratio, compressed image quality)

To get a more quantitative and experimental comparison, we use testing result from the Chen and Duan report, from which different images are compressed and parameters are compared.

2.2.1 Testing method, parameters and criteria

Testing method

The order of testing is FFT, SVD, Wavelet (db1), Wavelet (db2), and Wavelet (db4). Image with different compression ratios: about 80%, 90% and 98.5% is chosen, and according to these compressions choose the singular value of SVD.

For SVD, the Chen and Duan's test paper uses $m \cdot k + n \cdot k = (m+n) \cdot k$ memory places to store u_1 through u_k and $\lambda_1 v_1$ through $\lambda_k v_k$. Note that this is different from that in the image compression section in this paper. It is because in Chen and Duan assumption, singular values are stored together with right singular vector (v_i).

Three different quality levels of the compressed figures are obtained:

A+: recognizable from original one, **A**: acceptable, **A-**: can’t be accepted.

All the images in five methods are compressed with three quality levels, and their compression ratios and error ratios are calculated (Chen and Duan P.42).

Color image treatment

For pictures with color, first divide the picture into three layers: red, green and blue. Then we process the three layers respectively as the method used for gray pictures. After obtaining 3 compressed layers, combine them back into a single color picture (Chen and Duan P.36).

The error ratio

There are many error estimation parameters and the Chen and Duan report uses the L_2 norm and the PSNR, the peak signal-to-noise ratio.

L2 norm (Chen and Duan P.35)

Suppose that f(x, y) represents the original image, and g(x, y) is the compressed image, both of size are M×N, then:

$$L_2 \text{ norm of difference matrix} = \left\{ \frac{1}{NM} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left| g(x,y) - f(x,y) \right|^2 \right\}^{1/2}$$

(Note that it runs from 0 to M-1 instead of 1 to M, which is the convention used in the Fourier basis)

PSNR (Chen and Duan P.90)

The PSNR is defined in this way:

$$PSNR = 20 \cdot \log_{10}(\frac{MAX_I}{L_2})$$

MAXI is the maximum possible pixel value of the image.

Calculation of error ratio

The procedure to obtain the error ratio of image compression is as follows: First calculate the difference matrix between the original image and the compressed image by subtraction. Then calculate the L2-norm of the difference matrix and normalize it with the L2-norm of the original image. The result is the error rate of the image compression (Chen and Duan P.36).

For SVD, the calculation of error ratio is different from that in the image compression section. Here Chen and Duan (P.39) use spectral norm induced by L2 norm: $\|A\|_2 = \sqrt{\lambda_{\max}(A^* A)}$

For approximation matrix A_k of rank k,

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

The L_2 norm of the difference matrix is:

$$\|A - A_k\|_2 = \left\| \sum_{i=k+1}^{n} \sigma_i u_i v_i^T \right\|_2 = \sigma_{k+1} \text{(largest singular value)}$$

So the error ratio is:

$$\frac{\|A-A_k\|_2}{\|A\|_2}=\frac{\sigma_{k+1}}{\sigma_1}$$

Note that this is different from the error ratio defined by Frobenius norm $\left(\|A\|_F=\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}=\sqrt{\sum_{i=1}^{\min\{m,n\}}\sigma_i^2}\right)$ from which I use to measure difference in the image compression section. Both are good indicator of image compression accuracy.

<u>Comparison of the error ratio defined by Spectral norm and Frobenius norm</u>

For the error ratio measured by the spectral norm and Frobenius norm, an inequality relationship can be obtained:

"Determine m,k such that

$$f(m,k)\sqrt{\frac{\sum_{j=k+1}^{m}\sigma_j^2}{\sum_{j=1}^{m}\sigma_j^2}}\le\frac{\sigma_{k+1}}{\sigma_1}\le g(m,k)\sqrt{\frac{\sum_{j=k+1}^{m}\sigma_j^2}{\sum_{j=1}^{m}\sigma_j^2}}\text{ ,,}$$

<u>Proof</u>

First, note the following relationship:

For a given m, $\frac{\sigma_{k+1}}{\sigma_1}\le\sqrt{\frac{\sum_{j=k+1}^{m}\sigma_j^2}{\sum_{j=1}^{m}\sigma_j^2}}$ for small k; as k increases (i.e we keep more singular values in the approximation matrix), $\frac{\sigma_{k+1}}{\sigma_1}\ge\sqrt{\frac{\sum_{j=k+1}^{m}\sigma_j^2}{\sum_{j=1}^{m}\sigma_j^2}}$. We denote it as $S_e \le F_e$ and $S_e \ge F_e$. Therefore the error ratio defined by spectral norm is smaller than that defined by Frobenius norm for high compressed image while the converse happens for low compressed image.

Determination of $g(m,k)$ *for the right inequality.*

For $\frac{\sigma_{k+1}}{\sigma_1} \le g(m,k)\sqrt{\frac{\sum_{j=k+1}^{m}\sigma_j^2}{\sum_{j=1}^{m}\sigma_j^2}}$ to hold for all m and k, equivalently we need

$$\frac{\frac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\sum_{j=k+1}^{m}\sigma_j^2 \Big/ \sum_{j=1}^{m}\sigma_j^2}} \le g(m,k) \quad \text{for all m and k.}$$

The determination of such $g(m,k)$ can be achieved by finding the maximum value of $\frac{\frac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\sum_{j=k+1}^{m}\sigma_j^2 \Big/ \sum_{j=1}^{m}\sigma_j^2}}$, i.e, $\max\left\{\frac{\frac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\sum_{j=k+1}^{m}\sigma_j^2 \Big/ \sum_{j=1}^{m}\sigma_j^2}}\right\}$

N.B. $$\frac{\frac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\sum_{j=k+1}^{m}\sigma_j^2 \Big/ \sum_{j=1}^{m}\sigma_j^2}}$$

$$\leq \frac{\frac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\frac{\sigma_{k+1}^2+0+\cdots+0}{\sigma_1^2+\cdots+\sigma_1^2+\sigma_{k+1}^2+0+\cdots+0}}}$$

$$= \frac{\frac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\frac{\sigma_{k+1}^2}{k\sigma_1^2+\sigma_{k+1}^2}}}$$

$$= \sqrt{\frac{\sigma_{k+1}^2}{\sigma_1^2}\left(\frac{k\sigma_1^2+\sigma_{k+1}^2}{\sigma_{k+1}^2}\right)}$$

$$= \sqrt{k+\frac{\sigma_{k+1}^2}{\sigma_1^2}}$$

$$\leq \sqrt{k+1}$$

For $g(m,k)$ to hold for all m,k, we should take $g(m,k)=\sqrt{k+\frac{\sigma_{k+1}^2}{\sigma_1^2}}$ or

in terms of only m,k, $g(m,k)=\sqrt{k+1}$

Determination of $f(m,k)$ *for the left inequality*

For $f(m,k)\sqrt{\frac{\sum_{j=k+1}^{m}\sigma_j^2}{\sum_{j=1}^{m}\sigma_j^2}} \leq \frac{\sigma_{k+1}}{\sigma_1}$ to hold for all m and k, equivalently we need

$$f(m,k) \leq \frac{\frac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\sum_{j=k+1}^{m}\sigma_j^2 \Big/ \sum_{j=1}^{m}\sigma_j^2}}$$ for all m and k.

The determination of such $f(m,k)$ can be achieved by finding the minimum value of $\dfrac{\dfrac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\sum\limits_{j=k+1}^{m}\sigma_j^2 \Big/ \sum\limits_{j=1}^{m}\sigma_j^2}}$,

$$\text{i.e,}\min\left\{\frac{\dfrac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\sum\limits_{j=k+1}^{m}\sigma_j^2 \Big/ \sum\limits_{j=1}^{m}\sigma_j^2}}\right\}$$

N.B
$$\frac{\dfrac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\sum\limits_{j=k+1}^{m}\sigma_j^2 \Big/ \sum\limits_{j=1}^{m}\sigma_j^2}}$$

$$\geq \frac{\dfrac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\dfrac{\sigma_{k+1}^2+\cdots+\sigma_{k+1}^2}{\sigma_1^2+\cdots+\sigma_k^2+\sigma_{k+1}^2+\cdots+\sigma_{k+1}^2}}}$$

$$\geq \frac{\dfrac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\dfrac{\sigma_{k+1}^2+\cdots+\sigma_{k+1}^2}{\sigma_1^2+\sigma_{k+1}^2\cdots+\sigma_{k+1}^2+\sigma_{k+1}^2+\cdots+\sigma_{k+1}^2}}}$$

$$= \frac{\dfrac{\sigma_{k+1}}{\sigma_1}}{\sqrt{\dfrac{(m-k)\sigma_{k+1}^2}{\sigma_1^2+(m-1)\sigma_{k+1}^2}}}$$

$$= \sqrt{\left(\frac{\sigma_{k+1}^2}{\sigma_1^2}\right)\left(\frac{\sigma_1^2+(m-1)\sigma_{k+1}^2}{(m-k)\sigma_{k+1}^2}\right)}$$

$$= \sqrt{\frac{1}{(m-k)}+\frac{(m-1)}{(m-k)}\left(\frac{\sigma_{k+1}^2}{\sigma_1^2}\right)}$$

$$\to \sqrt{\frac{1}{(m-k)}} \quad \text{as} \quad \sigma_{k+1}^2 \to 0$$

For $f(m,k)$ to hold for all m,k, we should take

$f(m,k)=\sqrt{\frac{1}{(m-k)}+\frac{(m-1)}{(m-k)}\left(\frac{\sigma_{k+1}^2}{\sigma_1^2}\right)}$ or in terms of only m,k,

$f(m,k)=\sqrt{\frac{1}{(m-k)}}$.

So the inequality is found out to be:

$$\sqrt{\frac{1}{(m-k)}}\sqrt{\frac{\sum_{j=k+1}^{m}\sigma_j^2}{\sum_{j=1}^{m}\sigma_j^2}}\le\frac{\sigma_{k+1}}{\sigma_1}\le\sqrt{k+1}\sqrt{\frac{\sum_{j=k+1}^{m}\sigma_j^2}{\sum_{j=1}^{m}\sigma_j^2}}$$

Point to note

1. The reason behind the setting of values (e.g. 0) in the denominator and nominator of the fraction is by the following fact:

For $d>0$,

$$\frac{x+d}{y+d}-\frac{x}{y}=\frac{y(x+d)-x(y+d)}{y(y+d)}=\frac{yd-xd}{y(y+d)}=\frac{(y-x)d}{y(y+d)}$$

For y>x, if we add a positive number to both the denominator and nominator of $\frac{x}{y}$, $\frac{x}{y}$ will become bigger.

2. One implication of the inequality is that Frobenius norm is better in error measurement. Frobenius norm is better in the sense that it includes more information about the compression. For example, for an identity matrix, if we truncate 1 singular value out, the error ratio defined by spectal norm is 100%(independent of how many singular values truncated

actually) while that of Frobenius norm is relatively smaller. In this case, Frobenius norm reflects more about the truncation.

2.2.2 General result

7 images Fingerprint, wood, fungus, MRI, bird, coil and Duan (travel picture) are compressed by 3 different methods.

FFT and Wavelet are both good methods and FFT works best for fingerprint, wood (vertical texture) and Fungus. But FFT yields high error ratio for some images.

Wavelet works best for the remaining images and yields low error ratio for all images.

SVD cannot yield the lowest error ratio for 7 images but can yield lower error ratio than FFT for some images (Bird, Coil and Duan).

At the end of the report (Chen and Duan P.101), Chen and Duan used another error ratio (PSNR). The higher the PSNR means higher image quality. It turns out that the comparison result is similar for each image.

2.2.3 Comparison of 3 compression methods based on result

SVD

Advantages

1. (Chen and Duan P.79) Wavelet and FFT stop compressing the image beyond a certain compression degree, but SVD can still compress a lot. It also compresses up to contour (Chen and Duan P.103). When only general shape is required, SVD does good job and at the same time

save a lot of memory.

2. (Chen and Duan P.89) Mostly, the compression methods work better for the gray one then the color one. SVD has an opposite behavior; it works better for the color image than for the gray one.

<u>*Disadvantages*</u>

1. Image quality is not as good as Fourier and Wavelet.

<u>***Fourier transform***</u>

<u>*Advantages*</u>

1. DCT has directional property. It yields small error ratio for the texture image with a similar directional pattern, such as Wood image (Chen and Duan P.103).

<u>*Disadvantages*</u>

1. Limitation of compression degree exists. The compression always stops at certain ratio and cannot compress further (Chen and Duan P.104)

<u>***Wavelet transform***</u>

<u>*Advantages*</u>

1. Works well for all images and does not fluctuate in quality for different images.

<u>*Disadvantages*</u>

1. Limitation of compression degree exists. The compression always stops at certain ratio and cannot compress further (Chen and Duan

P.104)

3: Conclusion: When SVD will be more efficient

1. When ringing artifacts or blocking artifacts need to be avoided, SVD is a good choice.

2. When only general shape is needed to save memory, SVD can compress to contour with high compression ratio.

3. When compression speed is considered, SVD is a good choice.

4. Granted, the image quality of SVD is mostly worse than Fourier transform and Wavelet transform. However, there are some exceptions. When the image itself is not full-rank, then the last few singular values are zero and the last few terms in the outer product expansion can be truncated with no cost but save memory. Also, as stated in the special topic in the image compression section, we know that SVD is a good image compression technique when we apply it to an image that is near to a lower rank matrix (image). From the result from Chen and Duan paper, we can see that when the image are near to a lower rank image or possesses certain symmetry, SVD gives low error ratio.

Examples where SVD yields higher image quality

Out of the 7 images, SVD compression results in smaller error ratio than FFT in 2 images (Bird and coil).

We use both A+ images, because SVD usually performs bad when we compress to higher ratio.

Bird

W. CHEN AND W. DUAN

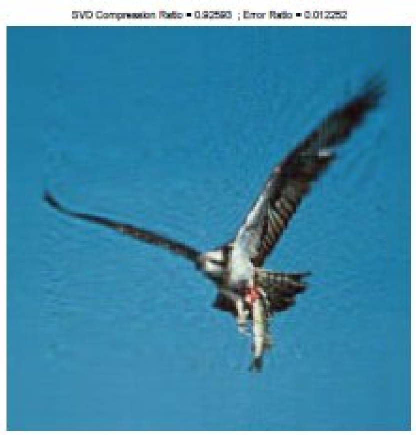


FIGURE 91. A+, bird, SVD, CR = 0.92593, ER = 0.012252

(SVD: ER=0.012252, FFT: ER=0.024011; image taken from Chen and Duan P.64)

For bird, part of the blue sky behind the bird spans the whole column/nearly whole column and constitutes near half of the image. As the columns representing blue sky are near or equal to each other, the whole image is indeed near to a low rank matrix where some columns are same as each other.

We can estimate the rank of the image matrix easily by observing the image. The columns representing and rows representing the sky are identical to each other.

rank $k \le \min\{p,q\}$, where p is the number of non-identical columns and q is the number of non-identical rows we observe by eye.

By noticing q, we can estimate the rank is about 80% of number of columns/rows.

<u>*Coil*</u>

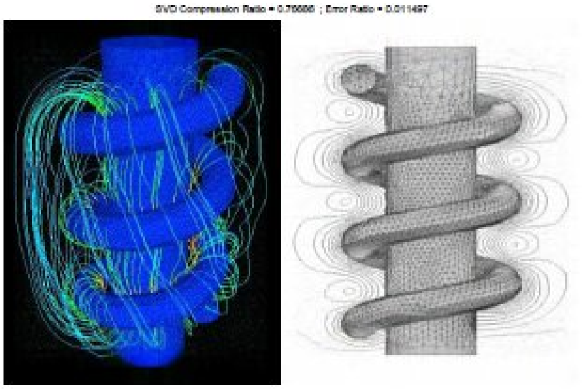


FIGURE 106. A+, coil, SVD, CR = 0.76686, ER = 0.011497

(SVD: ER=0.011497, FFT: ER=0.027834; image taken from Chen and Duan P.69)

For coil image, it combines the color part and grey part together. The symmetry argument is for the color part. We can observe that the color part is mainly about blue and black color. The black parts on the 2 sides are exactly identical (meaning the last few singular values of the image is exactly zero) and the blue parts in the middle have similar columns (most of their column elements are blue color).

Fingerprint

Another example worth looking at is the fingerprint image. Although SVD is still the worst, it is not so bad. (Generally, SVD yields error ratio 5-10 times larger than the other two methods)

W. CHEN AND W. DUAN

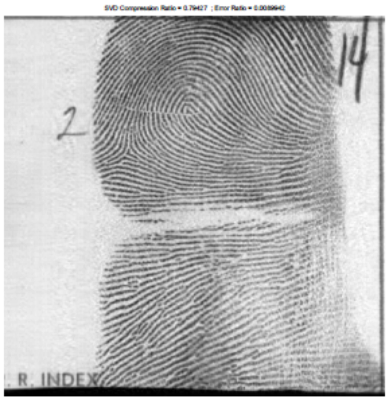


FIGURE 31. A+, FP, SVD, CR = 0.79427, ER = 0.0089942

(SVD: ER=0.0089942, FFT: ER= 0.0029535, haar’s wavelet: ER= 0.0050185; db2: ER= 0.004197; image taken from Chen and Duan P.44) The reason behind is again the column similarities in the left side of the image.

Points to note

It should be noted that, the linear dependencies or nearly linear dependencies between columns or rows do not necessarily imply that SVD is good for image compression. Additional information like the following is needed:

a) We have to note how many singular values (can be calculated from compression ratio) we decide to keep. Also, we have to note how large the (near) linear dependencies are (e.g. the number of columns similar to each other). Comparison between the two should be made, especially when we use the spectral norm to define the error ratio.

b) How close it is to linear dependencies, compared to the whole Frobenius norm or spectral norm of the original matrix. If there exists exact linear dependency like the coil, then the deletion of last singular value costs nothing.

c) It would be a mistake if linear dependence is considered from image, instead of from the matrix entries values. For example, in grayscale digital image, the difference of the entries values in the matrix for “black” and “white” is great. Therefore, a black pixel in 1 column and a white pixel in another would cause huge difference in terms of the Frobenius norm.

d) Although the effect is not huge, the use of different definition of error ratio may yield different conclusion. For example, the use of spectral norm and Frobenius norm in the error ratio calculation, especially when the error is small, may affect the conclusion. In general, the Frobenius norm is better to use in the error ratio.

e) Due to limited amount of time, it would be a future research project why 8×8 block based is generally used in Fourier transform instead of 64×1 column based. It is proposed by my supervisor that if 64×1 column based, the extent of blocking artifact, which is a serious disadvantage of Fourier transform, can be reduced. Different image testings need to be produced to confirm this hypothesis.

Part 4: Latent Semantic Indexing and other applications

Introduction

SVD also has applications in web searching. Latent semantic indexing is a mathematical application of SVD. It is a technique which has a particular strength in identifying the synonym and the "latent" semantics.

Traditional search

Term-document matrix

A term-document matrix is used in the traditional search technique.

Term-document matrix (D):

$$\text{D}= \quad \begin{array}{cc} & \begin{array}{ccc} \phantom{a_{11}} & d_j & \phantom{a_{1n}} \\ & \downarrow & \end{array} \\ t_i^T \rightarrow & \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \end{array}$$

$d_j = \begin{bmatrix} a_{1j} \\ \vdots \\ a_{mj} \end{bmatrix}$ is the document column vector

$t_i^T = \begin{bmatrix} a_{i1} \ldots a_{in} \end{bmatrix}$ is the term row vector

where the matrix elements a_{ij} represent the frequency of terms in documents.

Query vector

When we want to search a document, we form a query vector and compare it with the document vector.

Query vector (q):

$$q = \begin{pmatrix} q_1 \\ \vdots \\ q_n \end{pmatrix}$$

Indexing vector

Production of indexing vector(i):

We produce the indexing vector by the multiplying of transpose of term-document matrix(D^T) and query vector(q).

$$D^T q = i$$

where $i = \begin{pmatrix} i_1 \\ \vdots \\ i_n \end{pmatrix}$ is the indexing vector

and we order the webpages according to the order of component of the indexing vector.

Cosine angle method

The indexing vector will produce more accurate result if we consider further the norm of query and document vectors in the calculation. This can be achieved by using the cosine angle method.

In cosine angle method, we compare how close the document vector and query vector are by the cosine angle between them. i.e.

$$\cos\theta = \frac{q^T d_i}{\|q\|\|d\|}$$

The final indexing can be done by normalizing the document vectors in D and the query vector, and by taking the inner product between them. A modified indexing vector is produced:

$$D'^T q' = i'$$

where D' is the modified document term matrix in which the columns are normalized and q' is the normalized form of q.

i' is the modified indexing vector.

We order the webpages by the ordering of the modified indexing vector.

Near 1 means the document and query vector are close to each other

while near 0 means the two are not close.

The cosine angle method can eliminate the effect of:

1: Too many terms in the document vector (e.g. encyclopedia).

2: Too many terms in the query vector.

Latent semantic indexing (LSI)

Singular value decomposition

We decompose the term-document matrix D into three matrices:

$U, \Sigma, V,$

$s.t$

$D = U\Sigma V^T$

$$t_j^T \rightarrow \begin{array}{c} \begin{array}{ccc} & d_i & \\ & \downarrow & \end{array} \\ \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \end{array} = \left(\begin{pmatrix} \\ u_1 \\ \\ \end{pmatrix} \cdots \begin{pmatrix} \\ u_r \\ \\ \end{pmatrix} \right) \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_r \end{pmatrix} \begin{pmatrix} (& v_1^T &) \\ \vdots & & \\ (& v_r^T &) \end{pmatrix}$$

(wikipedia: latent semantic analysis)

Suppose we keep the first k singular values only, we get a rank k approximation to D. We denote this matrix by D_k, where $D_k = U_k \Sigma_k V_k^T$.

$$D_k = \left(\hat{t}_i^T\right) \rightarrow \left(\begin{pmatrix} \\ u_1 \\ \\ \end{pmatrix} \cdots \begin{pmatrix} \\ u_k \\ \\ \end{pmatrix} \right) \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_k \end{pmatrix} \begin{array}{c} \begin{array}{ccc} & \left(\hat{d}_j\right) & \\ & \downarrow & \end{array} \\ \begin{pmatrix} (& v_1^T &) \\ \vdots & & \\ (& v_k^T &) \end{pmatrix} \end{array}$$

After the rank k approximation, the term vectors are the rows of U_k while the document vectors are now the columns of V_k^T.

They are pseudo now as they are represented in lower dimension space and they are shorter than before.

<u>Computation of pseudo vectors:</u>

$$\hat{d}_j = \Sigma_k^{-1} U_k^T d_j$$
$$\hat{q} = \Sigma_k^{-1} U_k^T q$$

(wikipedia: latent semantic analysis)

<u>Effect of dimension reduction</u>

This example illustrates the effect of dimension reduction:

$$\{(\text{car}),(\text{truck}),(\text{flower})\} \rightarrow \{(1.3452 * \text{car}+0.2828*\text{truck}),(\text{flower})\}$$

(wikipedia: latent semantic analysis)

2 terms are combined to 1 term in the document vector and query vector.

<u>Example from www.miislita.com</u>

The query is gold silver truck and the “collection” consists of just three “documents”:

d1: Shipment of gold damaged in a fire

d2: Delivery of silver arrived in a silver truck

d3: Shipment of gold arrived in a truck

$$\begin{array}{ll}
\text{Terms} & \begin{array}{ccc} \text{d1} & \text{d2} & \text{d3} \end{array} \\
\begin{array}{l} \text{a} \\ \text{arrived} \\ \text{damaged} \\ \text{delievery} \\ \text{fire} \\ \text{gold} \\ \text{in} \\ \text{of} \\ \text{shipment} \\ \text{silver} \\ \text{truck} \end{array} &
\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \end{pmatrix}
\end{array}, \qquad q = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

<u>SVD results</u>

$$U = \begin{pmatrix} -0.4201 & 0.0748 & -0.0460 \\ -2.9995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{pmatrix}$$

$$V = \begin{pmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{pmatrix}$$

$$, \ V^T = \begin{pmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{pmatrix}$$

After the dimension reduction (for k=2)

$$U_k = \begin{pmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{pmatrix}, \Sigma_k = \begin{pmatrix} 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{pmatrix}$$

$$V_k = \begin{pmatrix} -0.4945 & 0.6492 \\ -0.6458 & -0.7194 \\ -0.5817 & 0.2469 \end{pmatrix}, V_k^T = \begin{pmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{pmatrix}$$

<u>Pseudo query vector:</u>

$$\hat{q} = \Sigma_k^{-1} U_k^T q$$

<u>Pseudo document vector:</u>

$$\hat{d}_j = \Sigma_k^{-1} U_k^T d_j$$

$sim(\hat{q}, \hat{d}_j) = sim(\Sigma_k^{-1} U_k^T q, \Sigma_k^{-1} U_k^T d_j)$,where sim means to take the cosine angle.

<u>Computation result</u>

Pseudo query vector

$$\hat{q} = \Sigma_k^{-1} U_k^T q$$

$$\hat{q} = \begin{pmatrix} \frac{1}{4.0989} & 0.0000 \\ 0.0000 & \frac{1}{2.3616} \end{pmatrix} \begin{pmatrix} -0.4201 & -0.2995 & -0.1206 & -0.1576 & -0.1206 & -0.2626 & -0.4201 & -0.4201 & -0.2626 & -0.3151 & -0.2995 \\ 0.0748 & -0.2001 & 0.2749 & -0.3046 & 0.2749 & 0.3794 & 0.0748 & 0.0748 & 0.3794 & -0.6093 & -0.2001 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -0.2140 \\ -0.1821 \end{pmatrix}$$

Pseudo document vectors:

$d_1(-0.4945, 0.6492)$
$d_2(-0.6458, -0.7194)$
$d_3(-0.5817, 0.2469)$

Cosine similarity in reduced space

$$sim(\hat{q}, \hat{d}_j) = \frac{\hat{q} \cdot \hat{d}_j}{\|\hat{q}\| \|\hat{d}_j\|}$$

$$sim(\hat{q}, \hat{d}_1) = \frac{(-0.2140)(-0.4945) + (-0.1821)(0.6492)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.4945)^2 + (0.6492)^2}} = -0.0541$$

$$sim(\hat{q}, \hat{d}_2) = \frac{(-0.2140)(-0.6458) + (-0.1821)(-0.7194)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.6458)^2 + (-0.7194)^2}} = 0.9910$$

$$sim(\hat{q}, \hat{d}_3) = \frac{(-0.2140)(-0.5817) + (-0.1821)(0.2469)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\sqrt{(-0.5817)^2 + (0.2469)^2}} = 0.4478$$

Conclusion

I conclude by listing the advantages and disadvantages of LSI.

Advantages of LSI

Traditional method cannot deal with synonym problem. LSI is able to do that.

Drawback of LSI

While LSI can do this:

$\{(\text{car}),(\text{truck}),(\text{flower})\} \rightarrow \{(1.3452 * \text{car+0.2828*truck}),(\text{flower})\}$ where (1.3452*car+0.2828*truck) component could be interpreted as "vehicle".

However:

It is likely that cases like

$\{(\text{car}),(\text{bottle}),(\text{flower})\} \to \{(1.3452 * \text{car}+0.2828*\textbf{bottle}),(\text{flower})\}$ will also occur.

(wikipedia: latent semantic analysis)

<u>Other SVD applications</u>

<u>Engineering</u>

Linear Dynamical system

EOF analysis(of ocean topography and climate system)

Signal processing and pattern recognition

Numerical weather prediction(Lanczos methods)

Use of SVD in detecting damage in structures at an early stage

<u>Arts:</u>

Using latent semantic indexing in facebook by Laura Devendorf

http://artfordorks.com/2011/07/facebook-singular-value-decomposition/

An artistic video of the concept of LSI

<u>Communications:</u>

Wireless communications

Compression of videos and images

<u>Mathematics and statistics:</u>

Representation of the range and null space of a matrix

Pseudoinverse

Numerical linear algebra

Low-rank matrix approximation

Solving Homogeneous linear equations

Total least squares minimization

Principal Component Analysis

Summary

To conclude, SVD is a significant theory and yields many mathematics results. There are already many applications of SVD are profound and in the future, especially in the growing digital world, many more SVD applications will arise. In developing the applications, it is important to understand the mathematical meaning of SVD, for example, the relationship between singular values and the linear dependency as in the image compression section.

Reference

1. Dr. N.K. Tsing, Chapter 5 of HKU math 2303 second semester 09-10 class lecture notes

2. Gilbert Strang, the Fundamental theorem of linear algebra, American Mathematical Monthly, Volume 100, Issue 9(Nov.,1993), 848-855

3. James Chen, Image Compression with SVD, 2000. Available from: http://fourier.eng.hmc.edu/e161/lectures/svdcompression.html

4. Wikipedia contributors. Singular value decomposition [Internet]. Wikipedia, The Free Encyclopedia; 2012 Apr 13, 07:46 UTC [cited 2012 Apr 19]. Available from: http://en.wikipedia.org/w/index.php?title=Singular_value_decomposition&oldid=487135458.

5. Wenjing Chen and Wei Duan, computational aspects of mathematical models in image compression

6. Gilbert strang video lecture Lec 26 MIT 18.06 Linear Algebra, Spring 2005

7. The Scientist and Engineer's Guide to Digital Signal Processing (Chapter 27: Data Compression) by Steven W. Smith, 1998. Available from: http://www.dspguide.com/pdfbook.htm

8. Wikipedia contributors. Wavelet [Internet]. Wikipedia, The Free Encyclopedia; 2012 Mar 22, 11:59 UTC [cited 2012 Mar 24]. Available

from: http://en.wikipedia.org/w/index.php?title=Wavelet&oldid=483350774.

9. Wikipedia contributors. Wavelet transform [Internet]. Wikipedia, The Free Encyclopedia; 2012 Mar 21, 10:35 UTC [cited 2012 Mar 24]. Available from: http://en.wikipedia.org/w/index.php?title=Wavelet_transform&oldid=483087546.

10. Wikipedia contributors. Ringing artifacts [Internet]. Wikipedia, The Free Encyclopedia; 2012 Mar 8, 15:32 UTC [cited 2012 Mar 24]. Available from: http://en.wikipedia.org/w/index.php?title=Ringing_artifacts&oldid=480847120.

11. joriki, Why is 8x8 matrix chosen for Discrete Cosine Transform? feedback stackexchange 2011. Available from: http://math.stackexchange.com/questions/32491/why-is-8x8-matrix-chosen-for-discrete-cosine-transform

12. David Austin, Image Compression: Seeing What's Not There, Grand Valley State University, 2012. Available from: http://www.ams.org/samplings/feature-column/fcarc-image-compression

13. James Chen, Image Compression with SVD, 2000. Available from: http://fourier.eng.hmc.edu/e161/lectures/svdcompression.html

14. Mi lslita**,** SVD-LSI pdf.

Available at: www.miislita.com

15. Barbara Rosario, Latent Semantic Indexing: An overview(2000)

16.Wikipedia contributors. Latent semantic analysis [Internet]. Wikipedia, The Free Encyclopedia; 2012 Apr 19, 01:29 UTC [cited 2012 Apr 23]. Available from: http://en.wikipedia.org/w/index.php?title=Latent_semantic_analysis&oldid=488098487.