Project Report

SVD-based Image Compression

Submitted by

Dhanush Sagar

Roll No: EE25BTECH11021

Under the guidance of

Prof. GVV Sharma

Department of Electrical engineering

Contents

Chapter 1

Summary of Strang's Video

In This lecture Prof. Gilbert Strang explains how any matrix A can be decomposed into three matrices:

$$A = U\Sigma V^T \tag{1.1}$$

where:

- U: orthogonal matrix (left singular vectors)
- Σ: diagonal matrix with singular values
- V: orthogonal matrix (right singular vectors)

Formate

he initially started taking a matrix A ,and wanted to find a matrix V which contains orthonormal vectors v_i(of rowspace) such that result with each vector will give rise to αu_i which(u) are again orthonormal(u) therefore forming a orthonormal matrix .this was his basic idea.this can be written as $AV = U\Sigma$ where Σ is diagonal matrix of alpha's.now since V is a orthogonal matrix multiplying V^T both side will give eq(1.1).

finding V and U

for finding matrix we can just do A^TA to get V and AA^T to get U we do these we get resultent as $V\sigma^2V^T$ and $U\sigma^2U^T$ .Seeing these squation and comparing with QVQ^T(informed in the begining of the class)we can say directly that V are eigen vectors of A^TA and σ^2 are eigen values of of the same matrix and same thing goes from U .From these we derive matrices V ,U and also Σ.

Other important thing—
sir also mentioned about one important thing about orthonormal vectors of V and U span the whole input space (for V its R^n), But for all vectors vectors doesnot contribute to the Matrix A,only vectors belonging to row space (V) and column space (U) of A contribute ,remaining vectors which are still orthonormal to input space but belongoing to nullspace of A does not contribute basically there α will be zero(interms of $A^T A$ there are the eigen vectors of 0 eigen value) we can skip these vectors but we are including to make a squared full ranked orthogonal matrix.

How i used this idea to compress the image
now if we use all the singulur values and vectors we will get the exact image but intead of that if we will just use vectors and singular values that will contribute the majority of the image ,it will form the exact image with less clarity which would be fine.(basically we will find singular vectors corresponding to largest singular value which will contribute most) this will the also decrease the rank of Matrix A (which depends on number of singular values u take)

$$A_k = U_k \Sigma_k V_k^T \tag{1.2}$$

Chapter 2

Explanation of the Implemented Algorithm

2.1 Mathematical Background

To compress the image, we retain only the top k singular values:

$$A_k = U_k \Sigma_k V_k^T \tag{2.1}$$

where U_k contains the first k columns of U, Σ_k the top $k \times k$ singular values, and V_k the first k columns of V.

The algoritham which i used here is Power iteration method to find eq 2.1.

We need to find the V as a eigen vectors of$A^T A$ as i told before , for this in this method we will use a trick .

we will take random unt vector and continue multiplying it with the $A^T A$ now this random vector can be written interms of sumation of all eigen vectors in the form $\sigma c_i v_1$ we know eigen vectors when multiplied will not chang the direction it will just extend in that direction .So all the vectors will get extended in that direction proportinal to there eigen values .

$$let \vec{v} = c_1\vec{v_1} + c_2\vec{v_2} + c_3\vec{v_3} + \cdots + c_r\vec{v_r} \tag{2.2}$$

$$A^T A\vec{v} = \lambda_1 c_1\vec{v_1} + \lambda_2 c_2\vec{v_2} + \lambda_3 c_3\vec{v_3} + \lambda_4 c_4\vec{v_4} + \cdots + \lambda_r c_r\vec{v_r} \tag{2.3}$$

Therefore, the eigen vector corresponding to maximum eigen value will ge extended maximum so there will be more tilting the vector in that direction .

Now doing this large number times makes the vector all most in the direction therefore changing into that eigen vector.

$$A^T A\vec{v}_1 \approx \lambda_1 \vec{v}_1 \tag{2.4}$$

Also i will add a point , saying that$A^T A$ is symmetric all eigen vector are orthogonal

now we to find $u_1 =$ just multiply $\vec{v_1}$ with A and then find norm of that vector.

$$A\vec{v}_1 \ = \sigma \vec{u}_1 \tag{2.5}$$

$$\vec{u}_1 = \vec{u}_1 / ||\vec{u}_1|| \tag{2.6}$$

σ_1 value wil be equal to norm of u_1 if u remember sir's video $Av = \sigma u$

$$\sigma_1 = ||\vec{u}_1|| \tag{2.7}$$

$$A_1 = u_1 \sigma_1 v_1^T \tag{2.8}$$

remember to subtract A for each k from original matrix so that after in next time you do iteration u will get next major eigen vector.

$$Anew = A - u_k \sigma_k v_k^T \tag{2.9}$$

to find final matrix we just do the formala

$$A_k = U_k \Sigma_k V_k^T \tag{2.10}$$

where k is the number of eigen vectors you choose .

2.2 Pseudocode

Listing 2.1: Pseudocode for SVD-based Image Compression

```
Input: Image matrix A,output matrix B ,row n, columns m,
   iteraations T,rank k
Output: void

1. crearting a loop to get k  vectors of V.
2.taking a random vector (each element less then one)
3.creating a loop for iteration.
4.inside the loop do A^TAv and divide by its norm and continue.
5.once u get v, do Av and divide by its norm to u.
6.the norm is the sigma for that vectors.
7.to find matrix A_new for this eigen vector we do u*sigma*v^t
8.now,subtract A_new from A to find the next vector
9 this A_new is only B.
```

Chapter 3

Algorithm Comparison and Choice

There are several approaches for solving for Svd:

1. **jacobiSVD):** this procees was trying to mutiply a rotation matrix with main matrix A and and diagonalize it this basically compression the whole information along the diagonal and also mutliplying the these rotation matrix with matrix V and U and therefore bringing it in UAV^T
   why i didnt ch0ose ;
   1.it is not good for large matrixes
   2.i didnt understand it properly

2. **Divide ad conquer Svd** this method include dividing the main matrix into @simpler matrix and then find svd for those simplier matrix again
   why i didnt choose it because
   it was looking complication
   and we we ned to store full matrix (since the matrix is big its hard to do it)

We chose **Power iteration** because:

- the proces is very easy and simple
- her we are not calculating the complete matrix just number of vector and singular values we want fior the given image.
- very use full for large matrixes since there is no requirement of storing the complete matrix.
- since images are very large , doing by thismethod is very fast

Chapter 4

Reconstructed Images for Different k

Below are reconstructed images using different values of k of image 1:

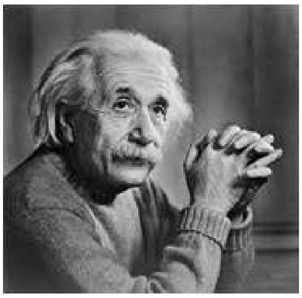
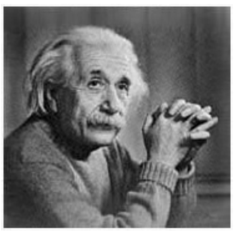


Figure 4.1: Main fig

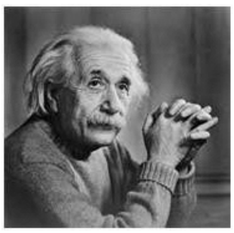
(a) k =5

(b) k =20

Figure 4.2



(a) k =50



(b) k =100

Figure 4.3

Below are reconstructed images using different values of k of image 2:

Figure 4.4: Main fig

(a) k =5

(b) k =20

Figure 4.5

(a) k =50

(b) k =100

Figure 4.6

Below are reconstructed images using different values of k of image 3:

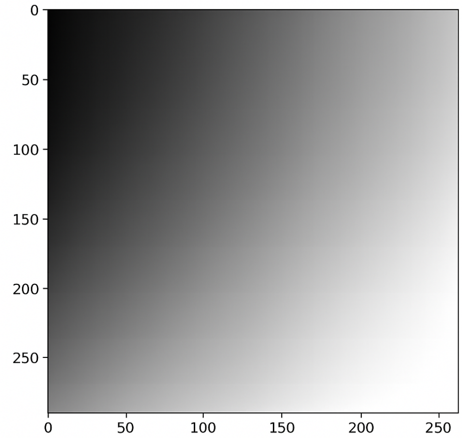


Figure 4.7: Main fig

(a) k =5

(b) k =20

Figure 4.8

(a) k =50

(b) k =100

Figure 4.9

Chapter 5

Error Analysis

We measure error using the Frobenius norm:

$$E(k) = ||A - A_k||_F \tag{5.1}$$

This measures how close the compressed image A_k is to the original A.

IMAGE1

k	Error
5	4714.572365
20	2126.564978
50	880.512100
100	164.780929

Table 5.1: Error values for different k for IMAGE1

IMAGE2

k	Error
5	20704.274560
20	10634.413443
50	6185.667722
100	3672.948772

Table 5.2: Error values for different k for IMAGE2

IMAGE3

k	**Error**
5	11146.309445
20	3808.193886
50	1159.896597
100	512.347669

Table 5.3: Error values for different k for IMAGE3

Chapter 6

Discussion and Reflections

the whole project was to use svd to compression .this provides a balance between storage efficiency and quality.Now lets talk on this
Trade-off:
1.when less number of k values are used ,its results in lesser storage space ,so image will be more compressed but at the cost of image quality.
2 when number of k increases quality of image , but the image compression deceases so storage occupied by the image increases . 3.i also want add n computation efficiency , since i am useing power iteration method ,it need not to store tyhe entire storage of image it only takes the singular values that has a greater part of contribution to image making image clear and light. implementstion : i am implementing the code via power iteration code ,I have alsready told the procedure of the implementation , i would just add one thing that we incraes the iteration the efficiency of this procedure increases by a great extent .therefore making image light aand clear. Overall, this project demonstrates the power of linear algebra in practical image processing applications.