Software Assignment - Image Compression Using Truncated SVD

EE25BTECH11049 - Sai Krishna Bakki

November 8, 2025

Introduction

A grayscale image can be represented as a matrix A $\in \mathbb{R}^{m \times n}$, where each entry of the matrix A a_{ij} corresponds to the pixel intensity (0 = black, 255 = white).

The **Singular Value Decomposition (SVD)** of the matrix A can be expressed as:

$$A = U\Sigma V^{\top} \tag{1}$$

By Taking the top k eigen values we have to reconstruct the matrix A as

$$A_k = U_k \Sigma_k V_k^{\top} \tag{2}$$

Singular Value Decomposition by Strang

Eigen value decomposition can be performed only on symmetric matrices.The decomposition which can be implemented on any matrix of arbitrary size is singular value decomposition

let A $\in \mathbb{R}^{m \times n}$ be a matrix of any size the **SVD** of A can be shown as

$$A = U\Sigma V^{\top} \tag{3}$$

where r is the rank of the matrix A

$$U \in \mathbb{R}^{m\times r} \text{is a matrix having its columns as orthonormal eigen vectors of } AA^\top \tag{4}$$

$$V \in \mathbb{R}^{n\times r} \text{is a matrix having its columns as orthonormal eigen vectors of } A^\top A \tag{5}$$

$$\Sigma^2 \in \mathbb{R}^{r\times r} \text{is a diagonal matrix having eigen values of } A^\top A \text{ in decreasing order} \tag{6}$$

$$\Sigma^2 = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r^2 \end{bmatrix} \tag{7}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ are the nonzero singular values of the matrix A.

Image to intensity matrix

The images in .jpg and in .png format can be coverted into intensity matrix by using a library in c language called **stb_image.h**. The command "STB_IMAGE_IMPLEMENTATION" must be included before the header. This tells the compiler to include the function implementations (actual code) from the header file.

Truncated SVD

There are many methods to perform truncated svd but the method which has less computational cost and more efficiency is **Randomized SVD using Power Iterations**

Randomized SVD using Power Iterations

Mathematical Background
let A $\in \mathbb{R}^{m\times n}$ be a matrix.Then,

$$A = U\Sigma V^\top \tag{8}$$

Computing full SVD for A is of order $O(mn^2)$ which is very expensive for large matrices like 512×512 and 1024×1024. So we use **randomization** to approximate the column space of A

Generate a Random matrix Ω

$$\Omega \in \mathbb{R}^{n \times l} \tag{9}$$
$$\Omega_{ij} \sim N(0, 1) \tag{10}$$
$$\text{where} l = k + p; p = \text{oversampling} \tag{11}$$

(10) means the entries of Ω are choosen such that average of all the entries is 0 and the variance is 1

$$Y = A\Omega \tag{12}$$

multiplying Ω with **A** will give a skinny matrix which is very useful to compute svd.

∴ Y contain almost the same subspace information as the top k singular vectors of A

Power Iterations

If singular values of A decay slowly then (11) may not be accurate so to make it more efficient we use power iterations

$$Y = (AA^\top)^{\mathrm{q}} A\Omega \tag{13}$$

where q is the number of power iterations typically q = 1 or 2 but q =2 handles very large matrices also very efficiently. q larger than 2 also gives nearly same accuracy as q = 2 . So let′s stick to q = 2

QR decomposition

Now we have to find the orthogonal basis of Y and make the columns to unit norm for that we can use QR decomposition based on Gram-schmidt algorithm

let

$$a_1, a_2 \ldots a_{\mathrm{k}} \text{ be the columns of the matrix Y} \tag{14}$$
$$q_1, q_2 \ldots q_{\mathrm{k}} \text{ be the orthonormal columns of the matrix Q} \tag{15}$$

$$q_1 = \frac{a_1}{||a_1||} \tag{16}$$
$$\tilde{q}_2 = a_2 - (a_2^\top q_1)\, q_1 \tag{17}$$
$$q_2 = \frac{\tilde{q}_2}{||\tilde{q}_2||} \tag{18}$$

we can generalize for q_k as

$$\tilde{q}_k = a_k - \sum_{j=1}^{k-1} \left(a_k \cdot q_j \right) q_j \tag{19}$$

$$q_k = \frac{\tilde{q}_k}{||\tilde{q}_k||} \tag{20}$$

Finally

$$Q = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \text{ and } Q \in \mathbb{R}^{m \times l} \tag{21}$$

where **Q** is the orthogonal matrix whose columns form a orthonormal basis for **Y** columns space.
Q approximates the span of top K singular values of A

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^{\top}\mathbf{A} \tag{22}$$

Projection
let′s project A into this k-dimensional subspace(Q)

$$\mathbf{B} = \mathbf{Q}^{\top}\mathbf{A} \tag{23}$$

$\mathrm{B} \in \mathbb{R}^{l \times n}$
B is a matrix whose dimension are kxn than A which retains the structure of A so it is easy to perform SVD on B than A
SVD

$$\mathbf{B} = \tilde{\mathbf{U}}\Sigma\mathbf{V}^{\top} \tag{24}$$

where

$$\tilde{U} \in \mathbb{R}^{l \times l} \text{is a matrix having it's columns as the top k eigen vectors of } \mathrm{BB}^{\top} \tag{25}$$

$$V \in \mathbb{R}^{n \times l} \text{is a matrix having it's columns as the top k eigen vectors of } \mathrm{A}^{\top}\mathrm{A} \tag{26}$$

$$\Sigma \in \mathbb{R}^{l \times l} \text{is a matrix having the top k singular values of matrix A} \tag{27}$$

Relation between U and $\tilde{U}$

$$B = Q^{\top}A \tag{28}$$

$$B^{\top} = A^{\top}Q \tag{29}$$

$$\implies BB^{\top} = (Q^{\top}A)(A^{\top}Q) \tag{30}$$

$$\implies BB^{\top} = Q^{\top}AA^{\top}Q \tag{31}$$

From (26)

$$BB^\top = \tilde{U}\Sigma\tilde{U}^\top \tag{32}$$
$$\implies Q^\top AA^\top Q = \tilde{U}\Sigma^2\tilde{U}^\top \tag{33}$$
$$\implies Q^\top AA^\top Q\tilde{U} = \tilde{U}\Sigma^2 \tag{34}$$

Multiply Q on left of (35)

$$QQ^\top AA^\top Q\tilde{U} = Q\tilde{U}\Sigma^2 \tag{35}$$

From (24) (37) can be written as

$$AA^\top Q\tilde{U} = Q\tilde{U}\Sigma^2 \tag{36}$$
$$AA^\top = Q\tilde{U}\Sigma^2\left(Q\tilde{U}\right)^{-1} \tag{37}$$

AS Q is orthogonal and $\tilde{U}$ is also orthogonal Q$\tilde{U}$ is also orthogonal

$$\left(Q\tilde{U}\right)^{-1} = \left(Q\tilde{U}\right)^\top \tag{38}$$

using (40) (39) can be written as

$$AA^\top = Q\tilde{U}\Sigma^2\left(Q\tilde{U}\right)^\top \tag{39}$$

As Σ^2 contains the squares of top K singular values the matrix Q$\tilde{U}$ should contain the eigen vectors corresponding to the top k values so we can conclude that

$$U = Q\tilde{U} \tag{40}$$

let A$_k$ be the matrix reconstructed using the top k singular values

$$\implies A_k = U\Sigma V^\top \tag{41}$$

From (42)

$$A_k = Q\tilde{U}\Sigma V^\top \tag{42}$$
$$\implies \mathbf{A}_k \mathbf{= QB} \tag{43}$$
$$Q \in \mathbb{R}^{m\times k} \text{ and } B \in \mathbb{R}^{k\times n} \tag{44}$$

Error Analysis

Frobenius norm
let A $\in \mathbb{R}^{m \times n}$ have SVD as

$$A = U\Sigma V^{\top} = \sum_{i=1}^{r} \sigma_i U_i V_i^T \tag{45}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ are the nonzero singular values of the matrix A. The rank-k truncated reconstruction of matrix A using the top k singular values is given by

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{46}$$

The Frobenius norm of the error $||A - A_k||$ is

$$||A - A_k||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} (A - A_k)_{ij}^2} \tag{47}$$

1 Code

```
    double frobeniusnorm(double **A, double **B, int m, int n) {
    double res = 0.0;
    for (int i = 0; i < m; ++i)
        for (int j = 0; j < n; ++j) {
            double d = A[i][j] - B[i][j];
            res += d * d;
        }
    return sqrt(res);
}
```

Intensity matrix to Image

Now we have to construct the image using the matrix which is reconstructed using the top k singular values. Based on input it decide either whether it is a .png or .jpg file based on that we can use the respective functions in stb_image_write.h

Trade-off between k, image quality, and compression.

- As the value of the k increases the Frobenious norm decreases

- As the value of k increases the quality and the sharpness of the image increases but the compression ratio decreases
- As the value of k tends to n the image obtained is almost similar to that of the original image

why only power iterations.?

When compared to other methods like jacobi transformation the major differences are :

- Methods like Jacobi Transformation Computes the full set of singular values and vectors where as in power iterations it Computes only a few (truncated) dominant singular values/vectors.
- Jacobi Transformation is often used for dense matrices and it is less efficent for large matrices but in power iterations it is highly efficient for large and sparse matrices.
- cost of the above algorithm to reconstruct the matrix using top k singular values is $O(mnk)$ where as jacobi transformation costs ($O(mk^2+nk^2)$) which is much more than the above algorithm cost
- Taking the above into consideration randomized SVD using power iterations is the best process

What if A is complex?

- The above algorithm and the computations done above is applicable only if A is a real matrix
- As in the above case the matrix A is intensity matrix there is no chance of A being Complex matrix but let′s check what to do if A is complex
- include<complex.h> in the code
- Replace all double with double complex in the above library
- Use conjugate transpose instead of simple transpose
- use conj() instead of dot product in the Gram_schmidt function
- Now there will be 2 parts in the matrix one is matrix of real component and other is a matrix of imaginary component