software assignment

AI25BTECH11035 - SUJAL RAJANI

SVD Geometric Meaning: Transforming v_1 to u_1

The fundamental relationship by SVD are :

$$Av_i = \sigma_i u_i \tag{1}$$

let r be the rank of A_{nm}

$$A\begin{pmatrix} v_1 & v_2 & \cdots & v_r \end{pmatrix} = \begin{pmatrix} u_1 & u_2 & \cdots & u_r \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{pmatrix} \tag{2}$$

where:

- V contains orthonomal vectors $v_1, v_2, \ldots, v_r$
- Σ contains singular values $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_r \geq 0$
- U contains orthonormal vectors $u_1, u_2, \ldots, u_r$

So the matrix A transforms the vector v_1 into vector u_1 whuch get scaled by the largest singular value σ_1:

$$Av_1 = \sigma_1 u_1 \tag{3}$$

This means:

- The direction v_1 in the input space is stretched by a factor of σ_1
- After stretching, the result lies along direction u_1 in the output space
- v_1 and u_1 are unit vectors and orthogonal to other singular vectors

$$AV = U\Sigma \tag{4}$$

The singular value decomposition (SVD) of a matrix A is

$$A = U\Sigma V^\top \tag{5}$$

Geometric interpretation

The SVD breaks down the action of A into three steps:

$$A = U\Sigma V^\top \tag{6}$$

1) $V^\top$ rotates the input vector into a coordinate system aligned with principal directions.
2) Σ is the diagonal matrix so it scales each axis by the singular values $\sigma_1, \sigma_2, \ldots$.
3) U rotates the result into the output space.

this all knowledge of geometry I get from the youtube channel visual kernel . Starting from the above,

$$A^\top A = V\Sigma^2 V^\top \tag{7}$$

Since $A^\top A$ is symmetric, it can be diagonalized as:

$$A^\top A = Q\Lambda Q^\top \tag{8}$$

where Q is the orthogonal matrix of eigenvectors, and Λ is the diagonal matrix of eigenvalues of $A^\top A$.

Thus, comparing both forms:

$$V = Q \quad \text{and} \quad \Sigma^2 = \Lambda \tag{9}$$

This means that each singular value σ_i is the square root of the corresponding eigenvalue λ_i of $A^\top A$:

$$\sigma_i = \sqrt{\lambda_i} \tag{10}$$

Explanation of the Implemented Algorithm (Power Block Iteration + QR)

Concept and Objective

The goal of our implemented algorithm is to compute the dominant singular values of a matrix $A \in \mathbb{R}^{m\times n}$ efficiently, without performing a full SVD decomposition.

The algorithm used is the Power Block Iteration method combined with QR orthogonalization in each iteration. This method generalizes the standard power method to a block of vectors, allowing us to compute multiple singular vectors simultaneously(which by the way is very efficient).

Math behind the algorithm

We already know that:

$$A^\top A v_i = \sigma_i^2 v_i \tag{11}$$

where v_i are the eigenvectors of $A^\top A$, and σ_i^2 are the eigenvalues of $A^\top A$.

The Power Block Iteration applies repeated multiplication by $A^\top A$ to a set of initial vectors V_0:

$$V_{k+1} = A^\top (A V_k) \tag{12}$$

Intuitively, each multiplication by $A^\top A$ amplifies the components of V_k in the direction of the dominant eigenvectors of $A^\top A$.

After several iterations, the columns of V_k become increasingly aligned with the top eigenvectors of $A^\top A$. This is due to the spectral property of symmetric matrices if $\lambda_1 > \lambda_2 > \cdots > \lambda_n$ are eigenvalues of $A^\top A$, then:

$$(A^\top A)^k V_0 \approx v_1 \lambda_1^k \tag{13}$$

which means the largest eigenvalue dominates as k increases.

Role of QR Decomposition

To prevent the vectors from collapsing into the same direction (numerical instability), we perform QR decomposition at each iteration:

$$[Q, R] = \mathrm{QR}(A^\top (A V_k)) \tag{14}$$

and then set:

$$V_{k+1} = Q \tag{15}$$

This ensures that the columns of V_k remain orthnormal, allowing multiple eigenvectors to be extracted simultaneously rather than converging to a single direction.

Convergence Intuition

Since $A^\top A$ is symmetric and positive semi-definite, its eigenvetors form an orthonormal basis. Repeated multiplication by $A^\top A$ amplifies the directions associated with the largest eigenvalues. Hence, the columns of V_k converge to the subspace spanned by the dominant eigenvectors of $A^\top A$.

After convergence:

$$AV \approx U\Sigma \tag{16}$$

and we can obtain:

$$\sigma_i = \|Av_i\|, \qquad u_i = \frac{Av_i}{\sigma_i} \tag{17}$$

which yields the leading singular vectors and singular values of A.

Why This Works : basically summary of what i am doing

1. $A^\top A$ is symmetric $\Rightarrow$ its eigenvectors are orthogonal.
2. The Power Iteration emphsizes the direction of the dominant eigenvalues.
3. QR orthognalization preserves numerical stability and orthogonality between iterates.
4. As iterations proceed, the column space of V_k converges to the eigenspace corresponding to the largest eigenvalues of $A^\top A$.

Thus, this method effectively computes the leading singular vectors of A without computing the full SVD. and in each we are given the number of k so we are going truncated SVD :

Truncated Singular Value Decomposition (TSVD)

Mathematical Definition

The full Singular Value Decomposition (SVD) of a matrix $A \in \mathbb{R}^{m\times n}$ is:

$$A = U\Sigma V^\top \tag{18}$$

The Truncated SVD (TSVD) keeps only the first k largest singular values and their corresponding singular vectors:

$$A_k = U_k \Sigma_k V_k^\top \tag{19}$$

where:

$$U_k = (u_1, u_2, \ldots, u_k) \in \mathbb{R}^{m\times k},$$
$$V_k = (v_1, v_2, \ldots, v_k) \in \mathbb{R}^{n\times k},$$
$$\Sigma_k = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k \end{pmatrix}$$

This means that A_k captures the most significant features or directions in A, corresponding to the k largest singular values $\sigma_1, \sigma_2, \ldots, \sigma_k$.

Mathematically, this can also be expressed as:

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^\top \tag{20}$$

Each term $\sigma_i u_i v_i^\top$ represents a rank:1 matrix contributing to A. The first few terms (largest σ_i) capture most of the energy (information) of the matrix.

Algorithmic Connection

In the implemented algorithm (Power Block Iteration with QR), we do not compute all singular values, but only the top-k ones. After convergence of V_k:

$$AV_k \approx U_k \Sigma_k \tag{21}$$

and hence:

$$A \approx U_k \Sigma_k V_k^\top \tag{22}$$

This truncated form reduces both computational cost and storage while preserving the essential structure of A.

Now it is the time to answer the big question that why I have chosen this algorithm over other :
In this project, we aim to perform image compression using the Truncated Singular Value Decomposition (TSVD) method. The objective is to approximate the image matrix $A \in \mathbb{R}^{m \times n}$ using only the top k singular values and vectors, where $k = 5, 10, 20, 50, 100$ are provided.

Instead of computing the full SVD, which has a high computational cost of $O(mn^2)$, we use the Power Block Iteration algorithm with QR decomposition at each step to efficiently compute only the leading singular components.

Advantages of power block algorithm for Image Compression

1. Computational :

We only compute the first k singular values and vectors, making the method much faster than the full SVD, especially for large image matrices.

2. Memory :

The algorithm avoids forming or storing the full $A^\top A$, which is beneficial when working with high-resolution images.

3.Numerical Stability:

The QR decomposition step prevents the iteration from collapsing into a single direction, maintaining orthogonality among the approximate singular vectors.

Why we are not using other algorithms?

1.Full SVD (Golub-Reinsch):

Very accurate but computationally heavy ($O(n^3)$). Not practical for large images (1024 x 1024) and hugher K.

2. Eigen-Decomposition of $A^\top A$:

this algorithm is more like brute force first we are finding each eigenvector then calculate sigma for each and then compare and finding the k largest .

3. Randomized SVD:

Although efficient, it introduces randomization and approximation variance. Power Block + QR gives deterministic, stable convergence(basically it doesn't perform well for lower k).

Error analysis

A. Frobenius Norm

The **Frobenius norm** of a matrix $A \in \mathbb{R}^{m \times n}$ is a measure of its overall magnitude and is defined as:

$$||A||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2} \tag{23}$$

It can also be expressed in terms of the singular values σ_i of A as:

$$||A||_F = \sqrt{\sum_{i=1}^{r} \sigma_i^2} \tag{24}$$

where r is the rank of A. The Frobenius norm is invariant under orthogonal transformations:

$$||A||_F = ||UAV^T||_F \tag{25}$$

for any orthogonal matrices U and V.

B. Error in Low-Rank Approximation

In image compression using Singular Value Decomposition (SVD) the original image matrix A is approximated by a rank-k matrix A_k:

$$A_k = U_k \Sigma_k V_k^T \tag{26}$$

where U_k and V_k contain the first k singular vectors, and Σ_k contains the top k singular values.

The approximation error is defined as the difference between the original image and its rank-k reconstruction:

$$E_k = A - A_k \tag{27}$$

The magnitude of this error is measured using the Frobenius norm:

$$||A - A_k||_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2} \tag{28}$$

This expression shows that the error depends entirely on the neglected singular values. As k increases, more of the dominant singular values are included in Σ_k, and the error decreases.

C. Relative Error

$$\text{Relative Error} = \frac{||A - A_k||_F}{||A||_F} \tag{29}$$

trade offs and reflections on implementation choice

The parameter k controls the rank of the approximation, i.e., how much information from the original image is preserved. Smaller k means stronger compression but lower quality; larger k means better quality but weaker compression.As we are going to increase k we are increasing the quality of images and we are doing less compression of image .

I. Pseudocode

Grayscale image file input.pgm, target rank k, number of iterations $ITER$ Compressed image file compressed.pgm, Frobenius error, and relative reconstruction error

1. **Step 1: Read and Normalize Image**
2. Read image data in binary `P5` format
3. Skip comment lines beginning with '#'
4. Obtain width (w), height (h), and max pixel value
5. Normalize pixel intensities: $img[i] = buf[i]/255.0$
6. **Step 2: Construct Matrix** A
7. Convert normalized datagreyscale into matrix A of size $(M \times N)$ where $M = h$, $N = w$
8. **Step 3: Compute** $A^T A$ 9.for$i = 1$ to N
10.for$j = 1$ to N
11. $C[i][j] = \sum_{k=1}^{M} A[k][i] \times A[k][j]$
12. end of loop
13.end of loop
14. **Step 4: Initialize Random Block Matrix**
15. Generate random matrix X of size $(N \times k)$
16. Normalize each column of X: $X_j = X_j/\|X_j\|_2$
17.**Step 5: Perform Block Power Iteration with QR**
18.for$iter = 1$ to $ITER$
19. $Y = (A^T A)X$
20. Apply Modified Gram-Schmidt QR decomposition:
21. $[Q, R] = QR(Y)$
22 .Update: $X = Q$
23.end of the loop
24 .**Step 6: Compute Singular Values**
25.for$j = 1$ to k
26. $v_j = X_j$
27. $tmp = Av_j$
28. $\sigma_j = \|tmp\|_2$
29.end of the loop
30.**Step 7: Compute Left Singular Vectors**
31.for$j = 1$ to k
32. $U_j = (Av_j)/\sigma_j$
33.end of the loop
34. **Step 8: Reconstruct Compressed Image**
35. $A_k = \sum_{r=1}^{k} \sigma_r U_r V_r^T$
36. Clamp pixel values of A_k to range $[0, 1]$
37. **Step 9: Compute Frobenius Norm and Error**
38. $\|A\|_F = \sqrt{\sum_i \sum_j A_{ij}^2}$
39.$\|A - A_k\|_F = \sqrt{\sum_i \sum_j (A_{ij} - (A_k)_{ij})^2}$
40. Relative Error $= \frac{\|A - A_k\|_F}{\|A\|_F}$
41.**Step 10: Write Output Image**
42. Scale A_k to 8-bit grayscale (0-255)
43. Write binary PGM file `compressed.pgm`
return the Compressed image, $\|A\|_F$, $\|A - A_k\|_F$, and relative error

Results: Visual Comparison
EINSTEIN

Fig. 1: Image 1: Original and reconstructed images for different k (left to right: original, $k = 5$, $k = 20$, $k = 50$, $k = 100$).

globe

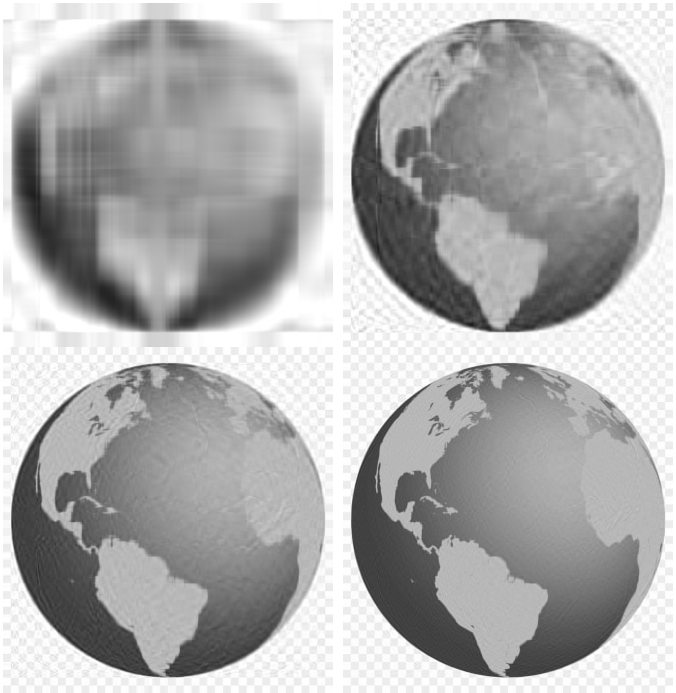

Fig. 2: Image 2: Original and reconstructed images for different k (left to right: original, $k = 5$, $k = 20$, $k = 50$, $k = 100$).

greyscale

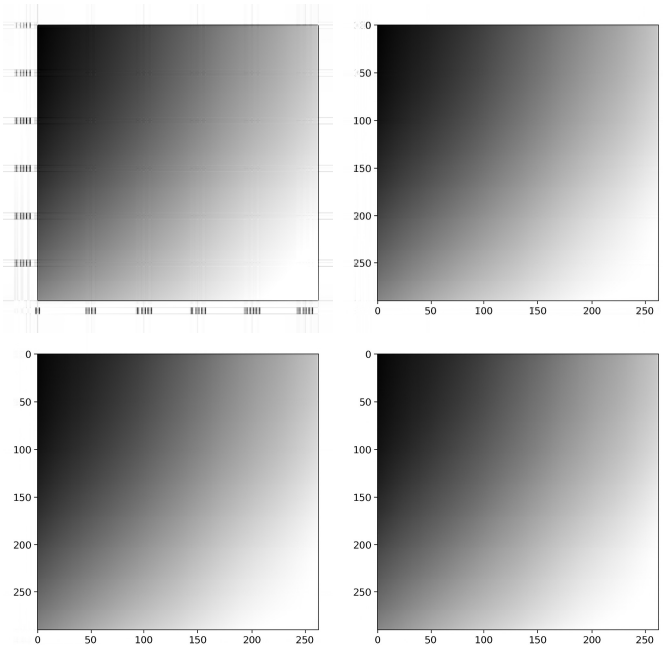

Fig. 3: Image 3: Original and reconstructed images for different k (left to right: original, $k = 5$, $k = 20$, $k = 50$, $k = 100$).