Course Project (Software) : Image Compression Using Truncated SVD

AI25BTECH11001 - ABHISEK MOHAPATRA

1) **Summary of Gibert Strang's video**
2) **Explanation of the implemented algorithm**
3) **Compare different algorithms and explain why did you choose the particular algorithm**
4) **Reconstructed images for different k**
5) **Error analysis**
6) **Discussion of trade-offs and reflections on implementation choice**

Summary of Gibert Strang's video

Each matrix $\mathbf{A}$ of order $m \times n$ can expressed in the form

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

where, $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrix of order $m \times m$ and $n \times n$, and

$$\mathbf{\Sigma} = diag\begin{pmatrix}\sigma_0 & \sigma_1 & \sigma_2 & ...\end{pmatrix}_{m\times n} \tag{2}$$

This can understand as basis vector of row space converted in column space

$$\mathbf{AV} = \mathbf{U\Sigma} \tag{3}$$

or,

$$\mathbf{A}\mathbf{v}_i = \sigma_i \mathbf{u}_i \tag{4}$$

And,$\mathbf{U}$ and $\mathbf{V}$ can be found by

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

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

where σ_i^2 are the eigen values of $\mathbf{A}\mathbf{A}^\top$ and $\mathbf{A}^\top\mathbf{A}$

Explanation of the Implemented Algorithm

Block Power Method

Block Power method replies on Power method for finding the eigenvector with the largest eigen vector.

Power Method

→ Each vector can be expressed in terms of as a linear combination of eigen vectors (as basis vector).

$$\mathbf{v} = \mu_0\mathbf{v}_0 + \mu_1\mathbf{v}_1 + \mu_2\mathbf{v}_2 + \mu_3\mathbf{v}_3 + ... + \mu_{n-1}\mathbf{v}_{n-1} \tag{7}$$

where $\mathbf{v}_i$ are eigen vector to matrix.

→ So, if we multiply the matrix with the vector the eigen values will be multiplied with each corresponding iron vectors and if we keep on multiplying this the eigen value of the eigen vectors with the largest eigen values will dominant over other vectors and the sequence will converge to a vector parallel to the eigen vector with the largest eigen value.

$$\mathbf{A}^m\mathbf{v} = \mu_0\lambda_0^m\mathbf{v}_0 + \mu_1\lambda_1^m\mathbf{v}_1 + \mu_2\lambda_2^m\mathbf{v}_2 + \mu_3\lambda_3^m\mathbf{v}_3 + ... + \mu_{n-1}\lambda_{n-1}^m\mathbf{v}_{n-1} \tag{8}$$

$$\mathbf{A}^m\mathbf{v} = \mu_0\lambda_0^m\left(\mathbf{v}_0 + \frac{\mu_1\lambda_1^m}{\mu_0\lambda_0^m}\mathbf{v}_1 + \frac{\mu_2\lambda_2^m}{\mu_0\lambda_0^m}\mathbf{v}_2 + \frac{\mu_3\lambda_3^m}{\mu_0\lambda_0^m}\mathbf{v}_3 + ... + \frac{\mu_{n-1}\lambda_{n-1}^m}{\mu_0\lambda_0^m}\mathbf{v}_{n-1}\right) \tag{9}$$

$$\lim_{m\to\infty}\mathbf{A}^m\mathbf{v} = \mu_0\lambda_0^m\mathbf{v}_0 \tag{10}$$

→ First we will initialise a random matrix $\mathbf{V}$ of n vectors then we will multiply $\mathbf{A}^\top\mathbf{A}$ ($\mathbf{A}$ = given matrix) to each of this vector and then Orthonormalized and continue this to a certain number of this step.

→ By applying this method, we will generate a Matrix with mutually perpendicular vectors so that these vectors are unique singular vectors and correspond to a unique singular value. This is our $\mathbf{V}$ matrix.

Initially:

$$\mathbf{V} := \begin{pmatrix}\mathbf{v_0} & \mathbf{v_1} & \mathbf{v_2} & ... & \mathbf{v_{n-1}}\end{pmatrix} \tag{11}$$

where $\mathbf{v}_i$ are randomly (distinct to each other) unit vectors.

Iteration upto m:

$$\mathbf{V} := \mathbf{A}^\top\mathbf{A}\mathbf{V} \tag{12}$$

$$\mathbf{V} := orthonorm(\mathbf{V}) \tag{13}$$

Finally:

$$let \quad \mathbf{B} = \mathbf{A}\mathbf{V} \tag{14}$$

$$U := orthonorm(\mathbf{B}) \tag{15}$$

And, let $\mathbf{C}$ be the product of the elementary operation performed on $\mathbf{B}$ to find $\mathbf{U}$.

$$\Sigma \approx C \quad as \quad m \to \infty \tag{16}$$

or, we take the diagonal elements of C as other vanishs as m increases.

Compare Different Algorithms and Explain Why Did You Choose the Particular Algorithm

In comparision to other algorithms, this algorithm is :

- very easy to understand
- easy to implement
- this algorithm has some base to basis of svd (obtaining svd from $\mathbf{A}^\top\mathbf{A}$ and $\mathbf{A}\mathbf{A}^\top$)

But there are some other trade of such that it is:

- very slow then others
- numerically instability
- requires higher value of K for better image

I used it as it will increase the understanding of matrices in general and easy to implement.

Reconstructed images for different k

Error analysis

Discussion of Trade-offs and Reflections on Implementation Choice

Using the power block method we can get our images compressed in like 5 to 10 minutes but there are a few drawbacks to this method first it is very slow in comparison to other. Secondly it produce noisy images at low values of the k. Thirdly, this algorithm is also not perfect as it doesn't give the actual singular values correctly and even make mistakes in some of the lower values of k.

Methods like Golub Khan method which is very fast efficient and in the mean while is stable and does it produce very good images at lower value of k and is widely used.