Optimal Hard Thresholding for SVD

Denoising and Rank Estimation in the Presence of Noise

SVD and Low-Rank Approximation

Standard Truncated SVD

The Eckart-Young theorem tells us that the best rank-k approximation of a matrix A is found by truncating its SVD, keeping only the top k singular values and vectors.

$$A_k = \sum_{i=1}^{k} \sigma_i \mathrm{u}_i \mathrm{v}_i^T$$

A Hidden Assumption

This assumes the matrix A is "clean". What happens when our data is corrupted by noise?

SVD and Low-Rank Approximation

Let's consider a matrix Y which is a noisy version of a true, low-rank signal matrix X:

$$Y = X + E$$

where E is a matrix of random noise (e.g., Gaussian noise).
Question: If we truncate the SVD of Y at rank-k, are we recovering X?

The Effect of Noise

Noise perturbs the singular values and vectors of the original signal.

- The singular values of the noisy matrix Y are not the same as the singular values of the clean matrix X.

The Effect of Noise

Noise perturbs the singular values and vectors of the original signal.

- The singular values of the noisy matrix Y are not the same as the singular values of the clean matrix X.
- Simply truncating the SVD of Y at rank k (the rank of X) will retain the noise captured in the first k singular components.

The Effect of Noise

Noise perturbs the singular values and vectors of the original signal.

- The singular values of the noisy matrix Y are not the same as the singular values of the clean matrix X.
- Simply truncating the SVD of Y at rank k (the rank of X) will retain the noise captured in the first k singular components.
- How do we distinguish which singular values correspond to the signal and which are just noise? How do we even know the true rank k?

The Goal

We need a principled way to separate the singular values that represent the underlying signal from those that are artifacts of the noise.

The Core Idea

Definition (Hard Thresholding)

Instead of just choosing a rank k, we define a threshold τ. We keep all singular values above this threshold and discard (set to zero) all singular values below it.

$$\hat{\sigma}_i = \begin{cases} \sigma_i & \text{if } \sigma_i \geq \tau \\ 0 & \text{if } \sigma_i < \tau \end{cases}$$

The denoised matrix is then reconstructed as $\hat{X} = \sum_i \hat{\sigma}_i u_i v_i^T$.

The Main Question

How do we choose the threshold τ in a non-arbitrary, optimal way?

Insight from Random Matrix Theory

The key insight comes from understanding the behavior of singular values of pure noise matrices.

Marchenko-Pastur Law

For a large $m \times n$ matrix with i.i.d. random entries, the distribution of its singular values is predictable. The largest singular value of a random noise matrix with variance σ^2 is not random—it converges to a specific upper bound.

Theorem (Donoho & Gavish, 2014)

For a large rectangular matrix corrupted by Gaussian noise, there is a sharp phase transition. The singular values of the underlying signal "pop out" above the bulk distribution of singular values from the noise. This allows for a theoretically optimal threshold.

The Optimal Hard Threshold

The Formula

For a noisy matrix $Y \in \mathbb{R}^{m \times n}$ (assuming $m \geq n$), the optimal hard threshold τ is given by:

$$\tau = \omega(\beta) \cdot \sigma_{\text{med}}$$

where:

- $\beta = m/n$ is the aspect ratio of the matrix.
- σ_{med} is the median singular value of Y, which serves as a robust estimator of the noise level.
- $\omega(\beta)$ is a value that depends only on the aspect ratio and can be approximated by a polynomial fit. A commonly used approximation is:

$$\omega(\beta) \approx 0.56\beta^3 - 0.95\beta^2 + 1.82\beta + 1.43$$

This threshold is optimal in the sense that it minimizes the squared error between the denoised matrix and the true signal matrix in the limit of large dimensions.

The Optimal Hard Threshold

Formula for square matrices and known noise

For a noisy square matrix $Y \in \mathbb{R}^{n \times n}$ then

$$\tau = (4/\sqrt{3})\sqrt{n}\gamma$$

where γ is the magnitude of the noise.

The Denoising Algorithm

Given a noisy data matrix $Y \in \mathbb{R}^{m \times n}$.

1. **Compute the SVD** of the noisy matrix:

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

The Denoising Algorithm

Given a noisy data matrix $Y \in \mathbb{R}^{m \times n}$.

1. **Compute the SVD** of the noisy matrix:

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

2. **Estimate Noise Level:** Find the median of the singular values: $\sigma_{\text{med}} = \text{median}(\text{diag}(\Sigma))$.

The Denoising Algorithm

Given a noisy data matrix $Y \in \mathbb{R}^{m \times n}$.

1. **Compute the SVD** of the noisy matrix:

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

2. **Estimate Noise Level:** Find the median of the singular values: $\sigma_{\text{med}} = \text{median}(\text{diag}(\Sigma))$.
3. **Calculate Optimal Threshold:** Compute the aspect ratio $\beta = m/n$ and find the corresponding $\omega(\beta)$. Calculate the threshold $\tau = \omega(\beta)\sigma_{\text{med}}$.

The Denoising Algorithm

Given a noisy data matrix $Y \in \mathbb{R}^{m \times n}$.

1. **Compute the SVD** of the noisy matrix:

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

2. **Estimate Noise Level:** Find the median of the singular values: $\sigma_{\text{med}} = \text{median}(\text{diag}(\Sigma))$.
3. **Calculate Optimal Threshold:** Compute the aspect ratio $\beta = m/n$ and find the corresponding $\omega(\beta)$. Calculate the threshold $\tau = \omega(\beta)\sigma_{\text{med}}$.
4. **Apply Thresholding:** Create a new diagonal matrix $\hat{\Sigma}$ where $\hat{\sigma}_i = \sigma_i$ if $\sigma_i \geq \tau$ and 0 otherwise. The number of non-zero singular values is the estimated rank $\hat{k}$.

The Denoising Algorithm

Given a noisy data matrix $Y \in \mathbb{R}^{m \times n}$.

1. **Compute the SVD** of the noisy matrix:

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

2. **Estimate Noise Level:** Find the median of the singular values: $\sigma_{\text{med}} = \text{median}(\text{diag}(\Sigma))$.
3. **Calculate Optimal Threshold:** Compute the aspect ratio $\beta = m/n$ and find the corresponding $\omega(\beta)$. Calculate the threshold $\tau = \omega(\beta)\sigma_{\text{med}}$.
4. **Apply Thresholding:** Create a new diagonal matrix $\hat{\Sigma}$ where $\hat{\sigma}_i = \sigma_i$ if $\sigma_i \geq \tau$ and 0 otherwise. The number of non-zero singular values is the estimated rank $\hat{k}$.
5. **Reconstruct Denoised Matrix:** Compute the denoised matrix: $\hat{X} = U \hat{\Sigma} V^T$.

Conclusion

- Simple SVD truncation is suboptimal for noisy data because it doesn't distinguish between signal and noise.

Conclusion

- Simple SVD truncation is suboptimal for noisy data because it doesn't distinguish between signal and noise.
- Optimal hard thresholding provides a mathematically principled way to denoise a matrix and estimate its true underlying rank.

Conclusion

- Simple SVD truncation is suboptimal for noisy data because it doesn't distinguish between signal and noise.
- Optimal hard thresholding provides a mathematically principled way to denoise a matrix and estimate its true underlying rank.
- The technique is grounded in powerful results from random matrix theory, which predict the behavior of singular values of noise.

Conclusion

- Simple SVD truncation is suboptimal for noisy data because it doesn't distinguish between signal and noise.
- Optimal hard thresholding provides a mathematically principled way to denoise a matrix and estimate its true underlying rank.
- The technique is grounded in powerful results from random matrix theory, which predict the behavior of singular values of noise.
- This method is widely applicable in fields like image processing, signal analysis, and machine learning, where data is often corrupted by noise.

SEE PYTHON NOTEBOOK!

The Johnson-Lindenstrauss Lemma

Statement of the Johnson-Lindenstrauss Lemma

Theorem (Johnson-Lindenstrauss Lemma)

Let $0 < \varepsilon < 1$. *For any set* X *of* N *points in* $\mathbb{R}^d$, *there exists a linear map* $f : \mathbb{R}^d \to \mathbb{R}^k$ *where* k *is on the order of*

$$k = O\left(\frac{\log N}{\varepsilon^2}\right)$$

such that for all pairs of points $\mathrm{u}, \mathrm{v} \in X$:

$$(1-\varepsilon)\left\|\mathrm{u}-\mathrm{v}\right\|_2^2 \leq \left\|f(\mathrm{u})-f(\mathrm{v})\right\|_2^2 \leq (1+\varepsilon)\left\|\mathrm{u}-\mathrm{v}\right\|_2^2$$

Key Idea

The lemma guarantees that any set of high-dimensional points can be projected into a much lower-dimensional space while approximately preserving the pairwise distances between them.

What Does It Really Mean?

The Core Idea: Preserving Geometry

Imagine you have a cloud of points in a very high-dimensional space (e.g., thousands of dimensions). It's impossible for us to visualize or work with directly.

What Does It Really Mean?

The Core Idea: Preserving Geometry

Imagine you have a cloud of points in a very high-dimensional space (e.g., thousands of dimensions). It's impossible for us to visualize or work with directly.

The Johnson-Lindenstrauss (JL) Lemma tells us something remarkable: we can randomly project this cloud of points onto a much, much lower-dimensional "shadow" space.

What Does It Really Mean?

The Core Idea: Preserving Geometry

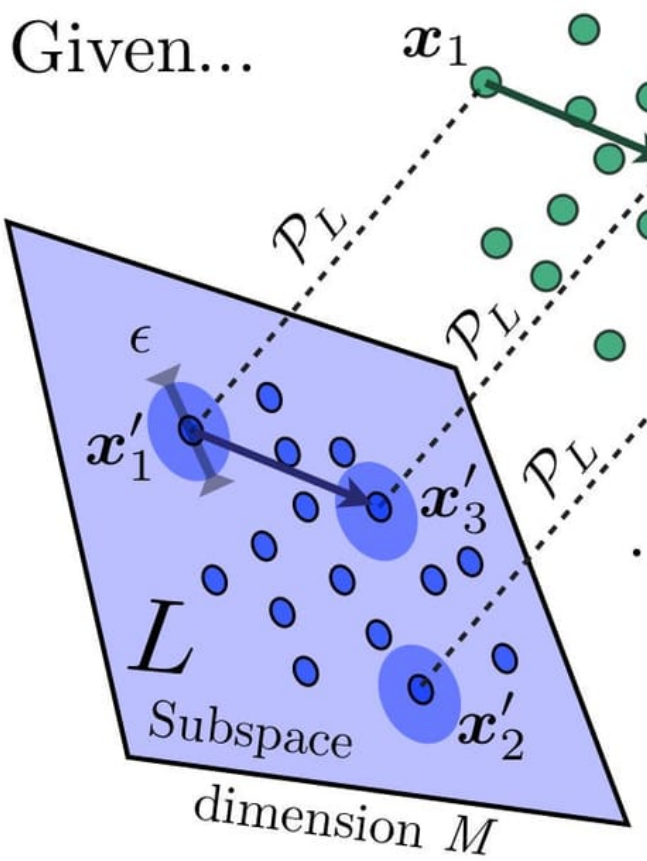
Imagine you have a cloud of points in a very high-dimensional space (e.g., thousands of dimensions). It's impossible for us to visualize or work with directly.

The Johnson-Lindenstrauss (JL) Lemma tells us something remarkable: we can randomly project this cloud of points onto a much, much lower-dimensional "shadow" space.

The magic is that the geometry of the point cloud—specifically, the distances between all pairs of points—is almost perfectly preserved in this shadow.

What Does It Really Mean?

Linear Dimensionality Reduction



... is there a L such that

$$\|\boldsymbol{x}'_i - \boldsymbol{x}'_j\| \simeq (1 \pm \epsilon)\|\boldsymbol{x}_i - \boldsymbol{x}_j\|$$

with $M = \dim L \ll N$?

The Surprising Part

The Target Dimension k is Independent of the Original Dimension d

The formula for the new dimension is:

$$k \approx \frac{\log(\text{Number of points})}{\text{Error}^2}$$

Notice what's missing: the original dimension, d!

The Surprising Part

The Target Dimension k is Independent of the Original Dimension d

The formula for the new dimension is:

$$k \approx \frac{\log(\text{Number of points})}{\text{Error}^2}$$

Notice what's missing: the original dimension, d!

- This means you can reduce data from 1 million dimensions to about 1,000 dimensions with the same success as reducing it from 10,000 dimensions.

The Surprising Part

The Target Dimension k is Independent of the Original Dimension d

The formula for the new dimension is:

$$k \approx \frac{\log(\text{Number of points})}{\text{Error}^2}$$

Notice what's missing: the original dimension, d!

- This means you can reduce data from 1 million dimensions to about 1,000 dimensions with the same success as reducing it from 10,000 dimensions.
- The new dimension depends only on how many points you have (N) and how much error you are willing to tolerate (ε).

The Surprising Part

The Target Dimension k is Independent of the Original Dimension d

The formula for the new dimension is:

$$k \approx \frac{\log(\text{Number of points})}{\text{Error}^2}$$

Notice what's missing: the original dimension, d!

- This means you can reduce data from 1 million dimensions to about 1,000 dimensions with the same success as reducing it from 10,000 dimensions.
- The new dimension depends only on how many points you have (N) and how much error you are willing to tolerate (ε).
- This is profoundly counter-intuitive but is the reason why the JL Lemma is so powerful for dealing with "big data".

The Randomized SVD (rSVD)

Efficient Low-Rank Matrix Approximation

Why Do We Need a New SVD Algorithm?

The Power of SVD

The Singular Value Decomposition is a cornerstone of numerical linear algebra, providing the best low-rank approximation of a matrix.

The Problem with Scale

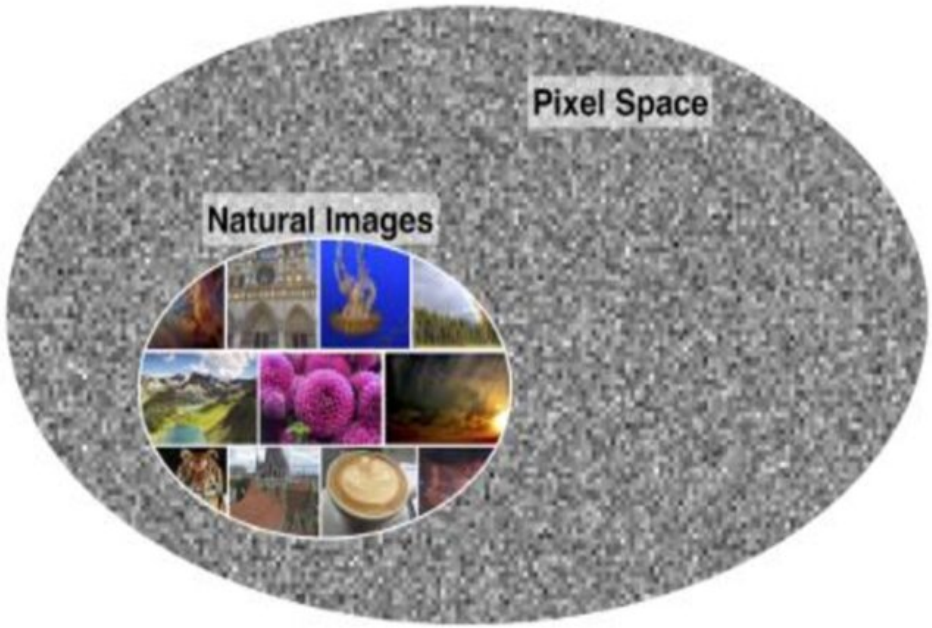
For very large matrices (e.g., millions of rows/columns), computing the full SVD is computationally prohibitive.

- Traditional algorithms have high polynomial complexity (e.g., $\mathcal{O}(mn^2)$).
- The matrix may not even fit into RAM, requiring slow out-of-core computations.

Why Do We Need a New SVD Algorithm?

Example

In many applications like image compression, collaborative filtering, or PCA on large datasets, we only need the first few dominant singular values and vectors.



Question: Can we find a good low-rank approximation without computing the full SVD?

Idea 1: Low-Rank Structure

The fundamental premise is that many large matrices are approximately low-rank. This means their information is concentrated in a low-dimensional subspace.

- The action of a matrix $A \in \mathbb{R}^{m \times n}$ on any vector can be described by its effect on the basis vectors of its range (column space).

Idea 1: Low-Rank Structure

The fundamental premise is that many large matrices are approximately low-rank. This means their information is concentrated in a low-dimensional subspace.

- The action of a matrix $A \in \mathbb{R}^{m \times n}$ on any vector can be described by its effect on the basis vectors of its range (column space).
- If A is approximately rank-k, we only need to find a basis for a k-dimensional subspace that captures most of the range of A.

The Challenge

How can we find this important subspace without analyzing the entire matrix A?

Idea 2: The Power of Random Projection

The Main Trick

Instead of analyzing A directly, we probe its structure with random vectors.

1. Generate a set of random vectors and collect them as columns in a matrix $\Omega \in \mathbb{R}^{n \times k}$, where k is our target rank.

Idea 2: The Power of Random Projection

The Main Trick

Instead of analyzing A directly, we probe its structure with random vectors.

1. Generate a set of random vectors and collect them as columns in a matrix $\Omega \in \mathbb{R}^{n \times k}$, where k is our target rank.
2. Form a new, "sketch" matrix Y by multiplying A with our random matrix:

$$Y = A\Omega \quad (Y \in \mathbb{R}^{m \times k})$$

Idea 2: The Power of Random Projection

The Main Trick

Instead of analyzing A directly, we probe its structure with random vectors.

1. Generate a set of random vectors and collect them as columns in a matrix $\Omega \in \mathbb{R}^{n \times k}$, where k is our target rank.
2. Form a new, "sketch" matrix Y by multiplying A with our random matrix:

$$Y = A\Omega \quad (Y \in \mathbb{R}^{m \times k})$$

3. The columns of Y are random linear combinations of the columns of A.

The Johnson-Lindenstrauss Lemma Intuition

With very high probability, the columns of Y span a subspace that captures the most important part of the range of A. The random projection preserves the essential geometry.

The rSVD Algorithm: A Two-Stage Process

The algorithm can be broken down into two main stages:

Stage A: Find the Subspace

Find an orthonormal matrix Q whose columns approximate the range of A.

Stage B: Decompose on the Subspace

Use Q to project A into a much smaller matrix, and then compute the SVD of this small matrix.

Algorithm Step-by-Step

Given a matrix $A \in \mathbb{R}^{m \times n}$ and a target rank k.

1. **Create a random sketch:** Generate a random Gaussian matrix $\Omega \in \mathbb{R}^{n \times k}$. Compute the sketch matrix $Y = A\Omega$.

Algorithm Step-by-Step

Given a matrix $A \in \mathbb{R}^{m \times n}$ and a target rank k.

1. **Create a random sketch:** Generate a random Gaussian matrix $\Omega \in \mathbb{R}^{n \times k}$. Compute the sketch matrix $Y = A\Omega$.
2. **Find an orthonormal basis:** Compute the "thin" QR decomposition of the sketch Y.

$$Y = QR$$

The columns of $Q \in \mathbb{R}^{m \times k}$ form an orthonormal basis for the approximate range of A.

Algorithm Step-by-Step

Given a matrix $A \in \mathbb{R}^{m \times n}$ and a target rank k.

1. **Create a random sketch:** Generate a random Gaussian matrix $\Omega \in \mathbb{R}^{n \times k}$. Compute the sketch matrix $Y = A\Omega$.
2. **Find an orthonormal basis:** Compute the "thin" QR decomposition of the sketch Y.

   $$Y = QR$$

   The columns of $Q \in \mathbb{R}^{m \times k}$ form an orthonormal basis for the approximate range of A.
3. **Project onto the subspace:** Project the original matrix A onto the low-dimensional basis defined by Q.

   $$B = Q^T A \quad (B \in \mathbb{R}^{k \times n})$$

   Note that $A \approx QQ^T A = QB$.

Algorithm Step-by-Step

Given a matrix $A \in \mathbb{R}^{m \times n}$ and a target rank k.

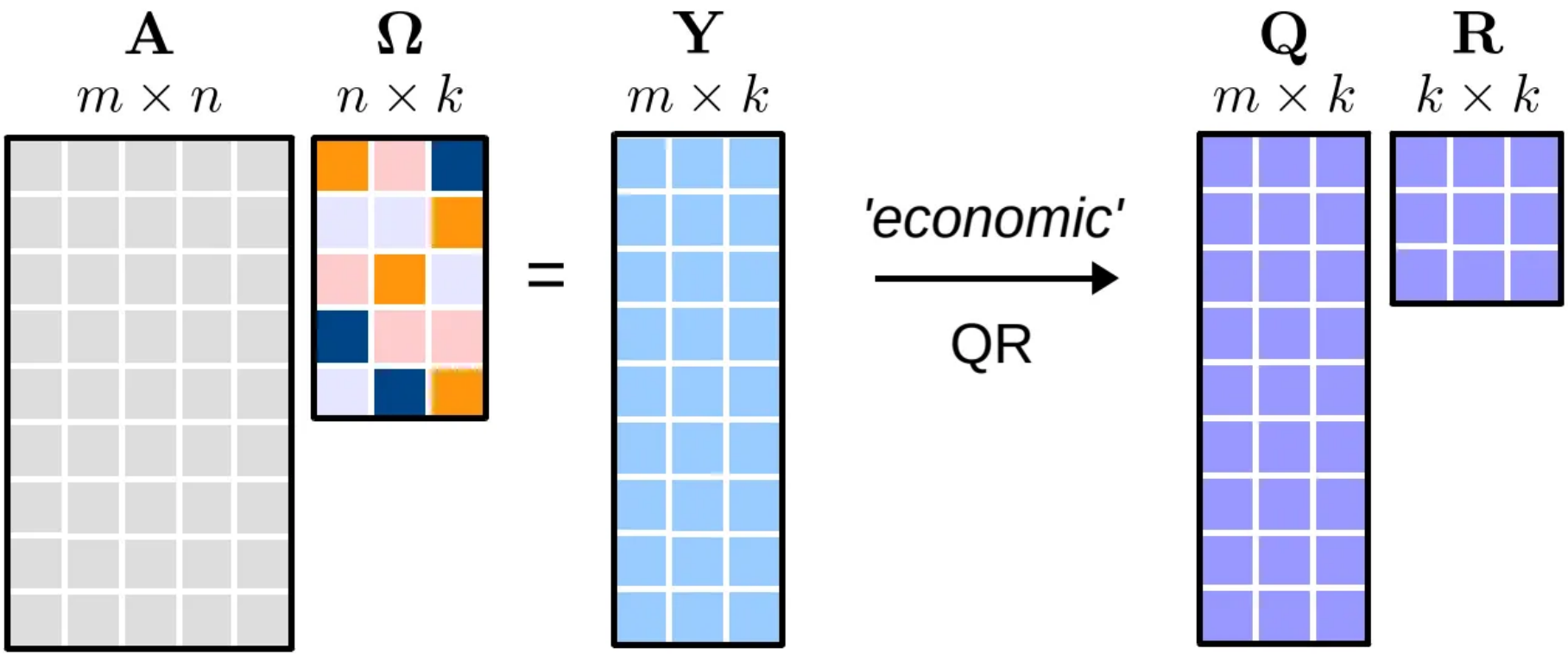
1. **Create a random sketch:** Generate a random Gaussian matrix $\Omega \in \mathbb{R}^{n \times k}$. Compute the sketch matrix $Y = A\Omega$.
2. **Find an orthonormal basis:** Compute the "thin" QR decomposition of the sketch Y.
$$Y = QR$$
The columns of $Q \in \mathbb{R}^{m \times k}$ form an orthonormal basis for the approximate range of A.
3. **Project onto the subspace:** Project the original matrix A onto the low-dimensional basis defined by Q.
$$B = Q^T A \quad (B \in \mathbb{R}^{k \times n})$$
Note that $A \approx QQ^T A = QB$.
4. **Compute SVD of the small matrix:** Compute the SVD of the small matrix B.
$$B = \tilde{U}\Sigma V^T$$

Algorithm Step-by-Step

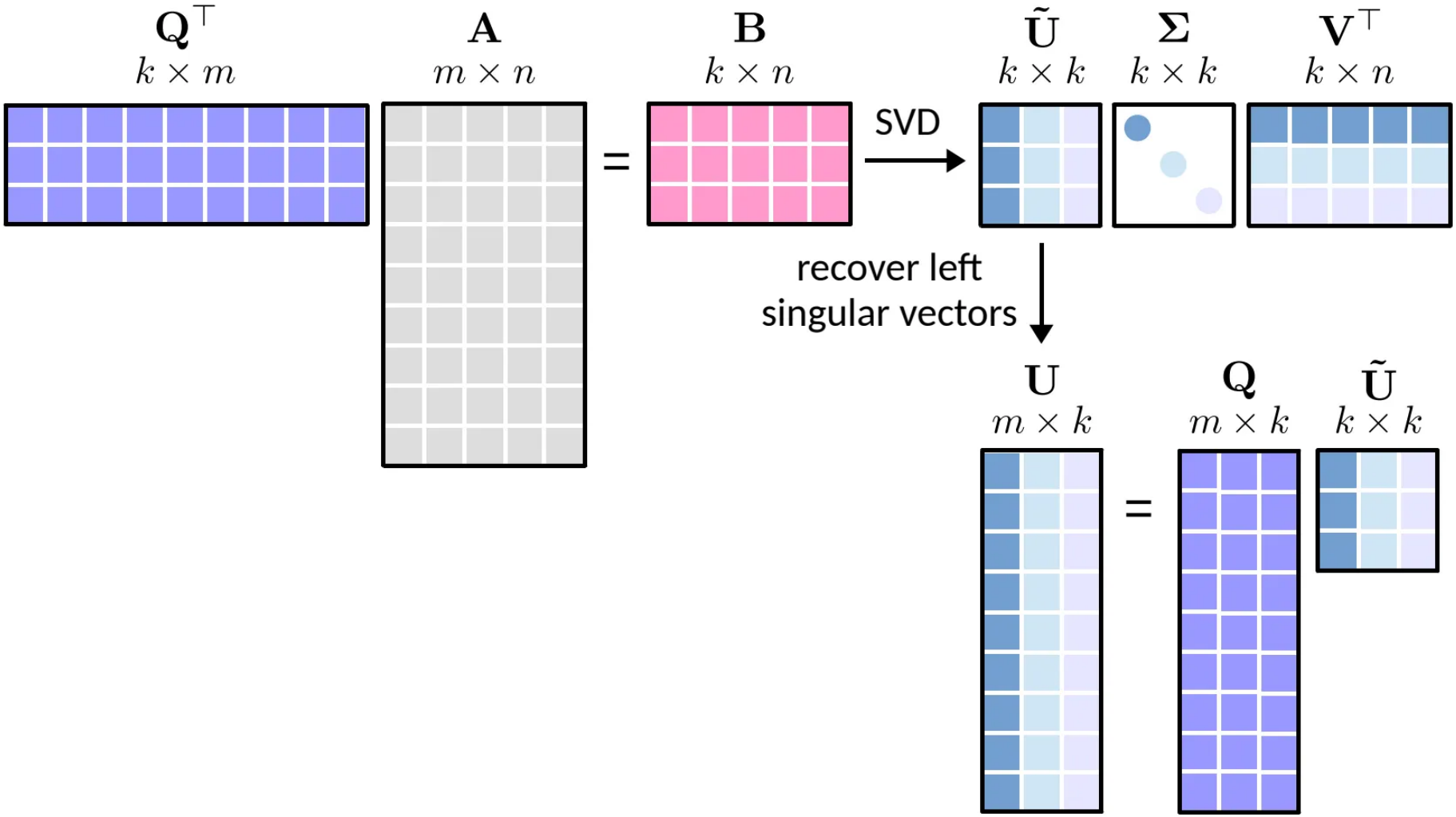
Given a matrix $A \in \mathbb{R}^{m \times n}$ and a target rank k.

1. **Create a random sketch:** Generate a random Gaussian matrix $\Omega \in \mathbb{R}^{n \times k}$. Compute the sketch matrix $Y = A\Omega$.
2. **Find an orthonormal basis:** Compute the "thin" QR decomposition of the sketch Y.
$$Y = QR$$
The columns of $Q \in \mathbb{R}^{m \times k}$ form an orthonormal basis for the approximate range of A.
3. **Project onto the subspace:** Project the original matrix A onto the low-dimensional basis defined by Q.
$$B = Q^T A \quad (B \in \mathbb{R}^{k \times n})$$
Note that $A \approx QQ^T A = QB$.
4. **Compute SVD of the small matrix:** Compute the SVD of the small matrix B.
$$B = \tilde{U}\Sigma V^T$$
5. **Reconstruct the final SVD:** Substitute back to get the final approximate SVD of A.
$$A \approx QB = Q(\tilde{U}\Sigma V^T) = (Q\tilde{U})\Sigma V^T$$
The approximate SVD of A is $U = Q\tilde{U}$, Σ, and V.

Graphical Representation (1)



Graphical Representation (2)



Why and When to Use rSVD

Key Advantages

- **Speed:** rSVD is significantly faster than traditional SVD for large, approximately low-rank matrices. It involves matrix multiplications which are highly optimized, and a much smaller SVD.
- **Scalability:** The algorithm can be adapted for matrices that do not fit in memory.
- **Accuracy:** It provides strong probabilistic guarantees on the quality of the approximation.

Important Considerations

- **Oversampling:** For better accuracy, the sketch size k is usually chosen slightly larger than the desired rank r (e.g., $k = r + p$, where p is a small oversampling parameter like 5 or 10).
- **Power Iterations:** Accuracy can be further improved by using a few power iterations ($Y = (AA^T)^q A\Omega$) to better capture the top singular vectors.

SEE PYTHON NOTEBOOK!

An Introduction to Matrix Norms

What is a Matrix Norm?

Definition

A matrix norm on the space of $m \times n$ matrices is a function $\|\cdot\| : \mathbb{R}^{m \times n} \to \mathbb{R}$ that satisfies the following properties for all matrices $A, B \in \mathbb{R}^{m \times n}$ and any scalar $\alpha \in \mathbb{R}$:

1. **Non-negativity:** $\|A\| \geq 0$, and $\|A\| = 0$ if and only if A is the zero matrix.

What is a Matrix Norm?

Definition

A matrix norm on the space of $m \times n$ matrices is a function $\|\cdot\| : \mathbb{R}^{m \times n} \to \mathbb{R}$ that satisfies the following properties for all matrices $A, B \in \mathbb{R}^{m \times n}$ and any scalar $\alpha \in \mathbb{R}$:

1. **Non-negativity:** $\|A\| \geq 0$, and $\|A\| = 0$ if and only if A is the zero matrix.
2. **Absolute Homogeneity:** $\|\alpha A\| = |\alpha| \, \|A\|$.

What is a Matrix Norm?

Definition

A matrix norm on the space of $m \times n$ matrices is a function $\|\cdot\| : \mathbb{R}^{m \times n} \to \mathbb{R}$ that satisfies the following properties for all matrices $A, B \in \mathbb{R}^{m \times n}$ and any scalar $\alpha \in \mathbb{R}$:

1. **Non-negativity:** $\|A\| \geq 0$, and $\|A\| = 0$ if and only if A is the zero matrix.
2. **Absolute Homogeneity:** $\|\alpha A\| = |\alpha| \, \|A\|$.
3. **Triangle Inequality:** $\|A + B\| \leq \|A\| + \|B\|$.

Sub-multiplicativity

For square matrices, a crucial additional property is often required:

- **Sub-multiplicativity:** $\|AB\| \leq \|A\| \, \|B\|$.

This links the norm to matrix multiplication.

The Frobenius Norm: $\|A\|_F$

Definition

The Frobenius norm is the most intuitive matrix norm. It is the square root of the sum of the absolute squares of its elements, analogous to the Euclidean norm for vectors.

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

Example

For the matrix $A = \begin{pmatrix} 1 & -2 \\ 3 & 0 \end{pmatrix}$:

$$\|A\|_F = \sqrt{1^2 + (-2)^2 + 3^2 + 0^2} = \sqrt{1 + 4 + 9 + 0} = \sqrt{14}$$

Note

The Frobenius norm is sub-multiplicative but is **not** an induced norm.

Frobenius Norm: Alternative Definitions

The Frobenius norm can also be expressed in two other important ways:

1. **In terms of the trace:** The squared Frobenius norm is the trace of $A^H A$ (or $A^T A$ for real matrices).

$$\|A\|_F^2 = \text{tr}(A^H A)$$

Frobenius Norm: Alternative Definitions

The Frobenius norm can also be expressed in two other important ways:

1. **In terms of the trace:** The squared Frobenius norm is the trace of $A^H A$ (or $A^T A$ for real matrices).

$$\|A\|_F^2 = \text{tr}(A^H A)$$

2. **In terms of singular values:** It is also the square root of the sum of the squares of the singular values (σ_i) of the matrix.

$$\|A\|_F = \sqrt{\sum_{i=1}^{\text{rank}(A)} \sigma_i^2}$$

Connection

These definitions highlight the deep connection between the norm of a matrix, its trace, and its singular value spectrum.

Proof: $\|A\|_F^2 = \mathrm{tr}(A^H A)$

- Let $C = A^H A$. The diagonal elements of C are given by $C_{jj} = \sum_{i=1}^{m} (A^H)_{ji} A_{ij}$.

Proof: $\|A\|_F^2 = \mathrm{tr}(A^H A)$

- Let $C = A^H A$. The diagonal elements of C are given by $C_{jj} = \sum_{i=1}^{m} (A^H)_{ji} A_{ij}$.
- Since $(A^H)_{ji} = \overline{A_{ij}}$, we have:

$$C_{jj} = \sum_{i=1}^{m} \overline{A_{ij}} A_{ij} = \sum_{i=1}^{m} |A_{ij}|^2$$

Proof: $\|A\|_F^2 = \mathrm{tr}(A^H A)$

- Let $C = A^H A$. The diagonal elements of C are given by $C_{jj} = \sum_{i=1}^{m} (A^H)_{ji} A_{ij}$.
- Since $(A^H)_{ji} = \overline{A_{ij}}$, we have:

$$C_{jj} = \sum_{i=1}^{m} \overline{A_{ij}} A_{ij} = \sum_{i=1}^{m} |A_{ij}|^2$$

- The trace of C is the sum of its diagonal elements:

$$\mathrm{tr}(A^H A) = \sum_{j=1}^{n} C_{jj} = \sum_{j=1}^{n} \sum_{i=1}^{m} |A_{ij}|^2$$

Proof: $\|A\|_F^2 = \mathrm{tr}(A^H A)$

- Let $C = A^H A$. The diagonal elements of C are given by $C_{jj} = \sum_{i=1}^{m} (A^H)_{ji} A_{ij}$.
- Since $(A^H)_{ji} = \overline{A_{ij}}$, we have:

$$C_{jj} = \sum_{i=1}^{m} \overline{A_{ij}} A_{ij} = \sum_{i=1}^{m} |A_{ij}|^2$$

- The trace of C is the sum of its diagonal elements:

$$\mathrm{tr}(A^H A) = \sum_{j=1}^{n} C_{jj} = \sum_{j=1}^{n} \sum_{i=1}^{m} |A_{ij}|^2$$

- This is exactly the definition of the squared Frobenius norm.

$$\mathrm{tr}(A^H A) = \|A\|_F^2$$

Proof: $\|A\|_F^2 = \sum \sigma_i^2$

- We start from the previous result: $\|A\|_F^2 = \text{tr}(A^H A)$.

Proof: $\|A\|_F^2 = \sum \sigma_i^2$

- We start from the previous result: $\|A\|_F^2 = \text{tr}(A^H A)$.
- A fundamental property of the trace is that it equals the sum of the eigenvalues of the matrix. Let λ_j be the eigenvalues of $A^H A$.

$$\text{tr}(A^H A) = \sum_{j=1}^{n} \lambda_j(A^H A)$$

Proof: $\|A\|_F^2 = \sum \sigma_i^2$

- We start from the previous result: $\|A\|_F^2 = \text{tr}(A^H A)$.
- A fundamental property of the trace is that it equals the sum of the eigenvalues of the matrix. Let λ_j be the eigenvalues of $A^H A$.

$$\text{tr}(A^H A) = \sum_{j=1}^{n} \lambda_j(A^H A)$$

- By definition, the eigenvalues of the matrix $A^H A$ are the squares of the singular values of A.

$$\lambda_j(A^H A) = \sigma_j(A)^2$$

Proof: $\|A\|_F^2 = \sum \sigma_i^2$

- We start from the previous result: $\|A\|_F^2 = \text{tr}(A^H A)$.
- A fundamental property of the trace is that it equals the sum of the eigenvalues of the matrix. Let λ_j be the eigenvalues of $A^H A$.

$$\text{tr}(A^H A) = \sum_{j=1}^{n} \lambda_j(A^H A)$$

- By definition, the eigenvalues of the matrix $A^H A$ are the squares of the singular values of A.

$$\lambda_j(A^H A) = \sigma_j(A)^2$$

- Combining these facts, we get:

$$\|A\|_F^2 = \sum_{j=1}^{\text{rank}(A)} \sigma_j^2$$

Frobenius Norm: Unitary Invariance

Property

The Frobenius norm is invariant under multiplication by unitary (or orthogonal for real matrices) matrices.

Theorem

Let $A \in \mathbb{C}^{m \times n}$, and let $Q \in \mathbb{C}^{m \times m}$ and $P \in \mathbb{C}^{n \times n}$ be unitary matrices (i.e., $Q^H Q = I$ and $P^H P = I$). Then:

$$\|QA\|_F = \|A\|_F \quad \text{and} \quad \|AP\|_F = \|A\|_F$$

Geometric Interpretation

This means that rotations and reflections do not change the "size" of the matrix as measured by the Frobenius norm.

Proof: Unitary Invariance

- We use the property $\|M\|_F^2 = \text{tr}(M^H M)$. Let's prove $\|QA\|_F = \|A\|_F$:

Proof: Unitary Invariance

- We use the property $\|M\|_F^2 = \text{tr}(M^H M)$. Let's prove $\|QA\|_F = \|A\|_F$:

$$\|QA\|_F^2 = \text{tr}((QA)^H (QA)) = \text{tr}(A^H Q^H QA)$$

Proof: Unitary Invariance

- We use the property $\|M\|_F^2 = \text{tr}(M^H M)$. Let's prove $\|QA\|_F = \|A\|_F$:

$$\|QA\|_F^2 = \text{tr}((QA)^H (QA)) = \text{tr}(A^H Q^H QA)$$

- Since Q is unitary, $Q^H Q = I$:

$$\text{tr}(A^H Q^H QA) = \text{tr}(A^H IA) = \text{tr}(A^H A) = \|A\|_F^2$$

Proof: Unitary Invariance

- We use the property $\|M\|_F^2 = \text{tr}(M^H M)$. Let's prove $\|QA\|_F = \|A\|_F$:

$$\|QA\|_F^2 = \text{tr}((QA)^H(QA)) = \text{tr}(A^H Q^H QA)$$

- Since Q is unitary, $Q^H Q = I$:

$$\text{tr}(A^H Q^H QA) = \text{tr}(A^H IA) = \text{tr}(A^H A) = \|A\|_F^2$$

- For the second part, $\|AP\|_F = \|A\|_F$, we use the cyclic property of the trace ($\text{tr}(XYZ) = \text{tr}(ZXY)$):

Proof: Unitary Invariance

- We use the property $\|M\|_F^2 = \text{tr}(M^H M)$. Let's prove $\|QA\|_F = \|A\|_F$:

$$\|QA\|_F^2 = \text{tr}((QA)^H(QA)) = \text{tr}(A^H Q^H QA)$$

- Since Q is unitary, $Q^H Q = I$:

$$\text{tr}(A^H Q^H QA) = \text{tr}(A^H IA) = \text{tr}(A^H A) = \|A\|_F^2$$

- For the second part, $\|AP\|_F = \|A\|_F$, we use the cyclic property of the trace ($\text{tr}(XYZ) = \text{tr}(ZXY)$):

$$\|AP\|_F^2 = \text{tr}((AP)^H(AP)) = \text{tr}(P^H A^H AP)$$

Proof: Unitary Invariance

- We use the property $\|M\|_F^2 = \text{tr}(M^H M)$. Let's prove $\|QA\|_F = \|A\|_F$:

$$\|QA\|_F^2 = \text{tr}((QA)^H(QA)) = \text{tr}(A^H Q^H QA)$$

- Since Q is unitary, $Q^H Q = I$:

$$\text{tr}(A^H Q^H QA) = \text{tr}(A^H IA) = \text{tr}(A^H A) = \|A\|_F^2$$

- For the second part, $\|AP\|_F = \|A\|_F$, we use the cyclic property of the trace ($\text{tr}(XYZ) = \text{tr}(ZXY)$):

$$\|AP\|_F^2 = \text{tr}((AP)^H(AP)) = \text{tr}(P^H A^H AP)$$

$$= \text{tr}(A^H APP^H)$$

Proof: Unitary Invariance

- We use the property $\|M\|_F^2 = \text{tr}(M^H M)$. Let's prove $\|QA\|_F = \|A\|_F$:

$$\|QA\|_F^2 = \text{tr}((QA)^H(QA)) = \text{tr}(A^H Q^H QA)$$

- Since Q is unitary, $Q^H Q = I$:

$$\text{tr}(A^H Q^H QA) = \text{tr}(A^H IA) = \text{tr}(A^H A) = \|A\|_F^2$$

- For the second part, $\|AP\|_F = \|A\|_F$, we use the cyclic property of the trace ($\text{tr}(XYZ) = \text{tr}(ZXY)$):

$$\|AP\|_F^2 = \text{tr}((AP)^H(AP)) = \text{tr}(P^H A^H AP)$$

$$= \text{tr}(A^H APP^H)$$

- Since P is unitary, $PP^H = I$:

$$\text{tr}(A^H APP^H) = \text{tr}(A^H A) = \|A\|_F^2$$

Induced (or Operator) p-Norms

Definition

An induced matrix norm is a norm that is defined in terms of a vector norm. Given a vector norm $\|\cdot\|_p$, the corresponding induced matrix norm is defined as the maximum "stretching factor" that the matrix applies to any non-zero vector.

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \sup_{\|x\|_p = 1} \|Ax\|_p$$

Important Cases

- $p = 1$: Max absolute column sum.
- $p = 2$: The Spectral Norm.
- $p = \infty$: Max absolute row sum.

Property

All induced norms are sub-multiplicative by definition.

Proof: Sub-multiplicativity of Induced Norms

- We want to show $\|AB\|_p \leq \|A\|_p \|B\|_p$.

Proof: Sub-multiplicativity of Induced Norms

- We want to show $\|AB\|_p \leq \|A\|_p \|B\|_p$.
- From the definition of the induced norm, for any vector x, we have:

$$\|M\mathrm{x}\|_p \leq \|M\|_p \|\mathrm{x}\|_p$$

Proof: Sub-multiplicativity of Induced Norms

- We want to show $\|AB\|_p \leq \|A\|_p \|B\|_p$.
- From the definition of the induced norm, for any vector x, we have:

$$\|M\mathrm{x}\|_p \leq \|M\|_p \|\mathrm{x}\|_p$$

- Let's apply this to the matrix product AB:

$$\|(AB)\mathrm{x}\|_p = \|A(B\mathrm{x})\|_p$$

Proof: Sub-multiplicativity of Induced Norms

- We want to show $\|AB\|_p \leq \|A\|_p \|B\|_p$.
- From the definition of the induced norm, for any vector x, we have:

$$\|M\mathrm{x}\|_p \leq \|M\|_p \|\mathrm{x}\|_p$$

- Let's apply this to the matrix product AB:

$$\|(AB)\mathrm{x}\|_p = \|A(B\mathrm{x})\|_p$$

- Letting $M = A$ and treating $B\mathrm{x}$ as a vector, we get:

$$\|A(B\mathrm{x})\|_p \leq \|A\|_p \|B\mathrm{x}\|_p$$

Proof: Sub-multiplicativity of Induced Norms

- We want to show $\|AB\|_p \leq \|A\|_p \|B\|_p$.
- From the definition of the induced norm, for any vector x, we have:

$$\|Mx\|_p \leq \|M\|_p \|x\|_p$$

- Let's apply this to the matrix product AB:

$$\|(AB)x\|_p = \|A(Bx)\|_p$$

- Letting $M = A$ and treating Bx as a vector, we get:

$$\|A(Bx)\|_p \leq \|A\|_p \|Bx\|_p$$

- Now apply the property again to $\|Bx\|_p$:

$$\|Bx\|_p \leq \|B\|_p \|x\|_p$$

Proof: Sub-multiplicativity of Induced Norms

- We want to show $\|AB\|_p \leq \|A\|_p \|B\|_p$.
- From the definition of the induced norm, for any vector x, we have:

$$\|Mx\|_p \leq \|M\|_p \|x\|_p$$

- Let's apply this to the matrix product AB:

$$\|(AB)x\|_p = \|A(Bx)\|_p$$

- Letting $M = A$ and treating Bx as a vector, we get:

$$\|A(Bx)\|_p \leq \|A\|_p \|Bx\|_p$$

- Now apply the property again to $\|Bx\|_p$:

$$\|Bx\|_p \leq \|B\|_p \|x\|_p$$

- Combining the inequalities:

$$\|(AB)x\|_p \leq \|A\|_p \|B\|_p \|x\|_p$$

Proof: Sub-multiplicativity of Induced Norms

- We want to show $\|AB\|_p \leq \|A\|_p \|B\|_p$.
- From the definition of the induced norm, for any vector x, we have:

$$\|Mx\|_p \leq \|M\|_p \|x\|_p$$

- Let's apply this to the matrix product AB:

$$\|(AB)x\|_p = \|A(Bx)\|_p$$

- Letting $M = A$ and treating Bx as a vector, we get:

$$\|A(Bx)\|_p \leq \|A\|_p \|Bx\|_p$$

- Now apply the property again to $\|Bx\|_p$:

$$\|Bx\|_p \leq \|B\|_p \|x\|_p$$

- Combining the inequalities:

$$\|(AB)x\|_p \leq \|A\|_p \|B\|_p \|x\|_p$$

The Spectral Norm: $\|A\|_2$

Definition

The spectral norm is the induced 2-norm, corresponding to the standard Euclidean vector norm. It is one of the most important matrix norms in linear algebra and numerical analysis.

$$\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2$$

Theorem

The spectral norm of a matrix A is equal to its largest singular value, $\sigma_{\max}(A)$.

$$\|A\|_2 = \sigma_1 = \sqrt{\lambda_{\max}(A^H A)}$$

where $\lambda_{\max}(A^H A)$ is the largest eigenvalue of the matrix $A^H A$.

The Spectral Norm: $\|A\|_2$

Geometric Interpretation

$\|A\|_2$ represents the largest possible stretching (or scaling) of a unit vector under the linear transformation defined by A.

Proof: $\|A\|_2 = \sigma_1$

- We start with the squared norm: $\|A\|_2^2 = \sup_{\|x\|_2=1} \|Ax\|_2^2$.

Proof: $\|A\|_2 = \sigma_1$

- We start with the squared norm: $\|A\|_2^2 = \sup_{\|x\|_2=1} \|Ax\|_2^2$.
- We can write $\|Ax\|_2^2 = (Ax)^H(Ax) = x^H A^H Ax$.

Proof: $\|A\|_2 = \sigma_1$

- We start with the squared norm: $\|A\|_2^2 = \sup_{\|x\|_2=1} \|Ax\|_2^2$.
- We can write $\|Ax\|_2^2 = (Ax)^H(Ax) = x^H A^H Ax$.
- Let $M = A^H A$. M is Hermitian and positive semi-definite. Its eigenvalues λ_i are real and non-negative, and it has an orthonormal basis of eigenvectors $\{v_i\}$.

Proof: $\|A\|_2 = \sigma_1$

- We start with the squared norm: $\|A\|_2^2 = \sup_{\|x\|_2=1} \|Ax\|_2^2$.
- We can write $\|Ax\|_2^2 = (Ax)^H(Ax) = x^H A^H Ax$.
- Let $M = A^H A$. M is Hermitian and positive semi-definite. Its eigenvalues λ_i are real and non-negative, and it has an orthonormal basis of eigenvectors $\{v_i\}$.
- By definition, $\lambda_i(A^H A) = \sigma_i(A)^2$. Let σ_1 be the largest singular value.

Proof: $\|A\|_2 = \sigma_1$

- We start with the squared norm: $\|A\|_2^2 = \sup_{\|x\|_2=1} \|Ax\|_2^2$.
- We can write $\|Ax\|_2^2 = (Ax)^H(Ax) = x^H A^H Ax$.
- Let $M = A^H A$. M is Hermitian and positive semi-definite. Its eigenvalues λ_i are real and non-negative, and it has an orthonormal basis of eigenvectors $\{v_i\}$.
- By definition, $\lambda_i(A^H A) = \sigma_i(A)^2$. Let σ_1 be the largest singular value.
- The expression $x^H Mx$ is the Rayleigh quotient. By the Rayleigh-Ritz theorem, its maximum value over all unit vectors x is the largest eigenvalue of M.

$$\sup_{\|x\|_2=1} x^H A^H Ax = \lambda_{\max}(A^H A) = \sigma_1^2$$

Proof: $\|A\|_2 = \sigma_1$

- We start with the squared norm: $\|A\|_2^2 = \sup_{\|x\|_2=1} \|Ax\|_2^2$.
- We can write $\|Ax\|_2^2 = (Ax)^H(Ax) = x^H A^H Ax$.
- Let $M = A^H A$. M is Hermitian and positive semi-definite. Its eigenvalues λ_i are real and non-negative, and it has an orthonormal basis of eigenvectors $\{v_i\}$.
- By definition, $\lambda_i(A^H A) = \sigma_i(A)^2$. Let σ_1 be the largest singular value.
- The expression $x^H Mx$ is the Rayleigh quotient. By the Rayleigh-Ritz theorem, its maximum value over all unit vectors x is the largest eigenvalue of M.

$$\sup_{\|x\|_2=1} x^H A^H Ax = \lambda_{\max}(A^H A) = \sigma_1^2$$

- Therefore, $\|A\|_2^2 = \sigma_1^2$. Taking the square root gives the final result:

$$\|A\|_2 = \sigma_1$$

Detailed Proofs of the Eckart-Young-Mirsky Theorem

Low-Rank Matrix Approximation

Statement of the Theorem

Setup

Let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank r. Its Singular Value Decomposition (SVD) is $A = U\Sigma V^T$, with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$.

Low-Rank Approximation A_k

For any integer $k < r$, the truncated SVD gives the rank-k matrix:

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

Statement of the Theorem

Theorem (Eckart-Young-Mirsky)

For any matrix $B \in \mathbb{R}^{m \times n}$ with $rank(B) \leq k$, A_k is the best approximation to A:

1. ***Spectral Norm:*** $\|A - A_k\|_2 = \min_{rank(B) \leq k} \|A - B\|_2 = \sigma_{k+1}$
2. ***Frobenius Norm:***
   $\|A - A_k\|_F = \min_{rank(B) \leq k} \|A - B\|_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}$

Proof for the Spectral Norm - Part 1: Error of A_k

Calculating the error $\|A - A_k\|_2$

The difference matrix is:

$$A - A_k = \sum_{i=1}^{r} \sigma_i u_i v_i^T - \sum_{i=1}^{k} \sigma_i u_i v_i^T = \sum_{i=k+1}^{r} \sigma_i u_i v_i^T$$

This is the SVD of $A - A_k$.

Proof for the Spectral Norm - Part 1: Error of A_k

Calculating the error $\|A - A_k\|_2$

The difference matrix is:

$$A - A_k = \sum_{i=1}^{r} \sigma_i u_i v_i^T - \sum_{i=1}^{k} \sigma_i u_i v_i^T = \sum_{i=k+1}^{r} \sigma_i u_i v_i^T$$

This is the SVD of $A - A_k$.
The largest singular value of this matrix is σ_{k+1}. By definition of the spectral norm (which is the largest singular value), we have:

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

Proof for the Spectral Norm - Part 1: Error of A_k

Goal

Now, we must show that for any other matrix B with $\text{rank}(B) \leq k$, the error is at least this large:

$$\|A - B\|_2 \geq \sigma_{k+1}$$

Proof for the Spectral Norm - Part 2: Subspace Intersection

- Let B be any matrix with $\text{rank}(B) \leq k$.

Proof for the Spectral Norm - Part 2: Subspace Intersection

- Let B be any matrix with $\text{rank}(B) \leq k$.
- The null space of B, $\mathcal{N}(B)$, has dimension:

$$\dim(\mathcal{N}(B)) = n - \text{rank}(B) \geq n - k$$

Proof for the Spectral Norm - Part 2: Subspace Intersection

- Let B be any matrix with $\text{rank}(B) \leq k$.
- The null space of B, $\mathcal{N}(B)$, has dimension:

$$\dim(\mathcal{N}(B)) = n - \text{rank}(B) \geq n - k$$

- Let S_{k+1} be the subspace spanned by the first $k+1$ right singular vectors of A:

$$S_{k+1} = \text{span}\{v_1, \ldots, v_{k+1}\}$$

This subspace has $\dim(S_{k+1}) = k + 1$.

Proof for the Spectral Norm - Part 2: Subspace Intersection

- Let B be any matrix with $\text{rank}(B) \leq k$.
- The null space of B, $\mathcal{N}(B)$, has dimension:

$$\dim(\mathcal{N}(B)) = n - \text{rank}(B) \geq n - k$$

- Let S_{k+1} be the subspace spanned by the first $k+1$ right singular vectors of A:

$$S_{k+1} = \text{span}\{v_1, \ldots, v_{k+1}\}$$

  This subspace has $\dim(S_{k+1}) = k + 1$.
- Since $\dim(\mathcal{N}(B)) + \dim(S_{k+1}) \geq (n-k) + (k+1) = n + 1 > n$, their intersection is non-trivial.

Proof for the Spectral Norm - Part 2: Subspace Intersection

- Let B be any matrix with $\text{rank}(B) \leq k$.
- The null space of B, $\mathcal{N}(B)$, has dimension:

$$\dim(\mathcal{N}(B)) = n - \text{rank}(B) \geq n - k$$

- Let S_{k+1} be the subspace spanned by the first $k+1$ right singular vectors of A:

$$S_{k+1} = \text{span}\{v_1, \ldots, v_{k+1}\}$$

  This subspace has $\dim(S_{k+1}) = k + 1$.
- Since $\dim(\mathcal{N}(B)) + \dim(S_{k+1}) \geq (n-k) + (k+1) = n + 1 > n$, their intersection is non-trivial.
- Therefore, there must exist a non-zero unit vector z such that:

$$z \in \mathcal{N}(B) \quad \text{and} \quad z \in S_{k+1}$$

Proof for the Spectral Norm - Part 3: The Lower Bound

- We know $Bz = 0$ and $z = \sum_{i=1}^{k+1} c_i v_i$ for some scalars c_i with $\sum c_i^2 = 1$.

Proof for the Spectral Norm - Part 3: The Lower Bound

- We know $Bz = 0$ and $z = \sum_{i=1}^{k+1} c_i v_i$ for some scalars c_i with $\sum c_i^2 = 1$.
- The squared norm of the error is:

$$\|A - B\|_2^2 \geq \|(A - B)z\|_2^2 = \|Az\|_2^2$$

Proof for the Spectral Norm - Part 3: The Lower Bound

- We know $Bz = 0$ and $z = \sum_{i=1}^{k+1} c_i v_i$ for some scalars c_i with $\sum c_i^2 = 1$.
- The squared norm of the error is:

$$\|A - B\|_2^2 \geq \|(A - B)z\|_2^2 = \|Az\|_2^2$$

- We compute Az:

$$Az = \left(\sum_{j=1}^{r} \sigma_j u_j v_j^T \right) \left(\sum_{i=1}^{k+1} c_i v_i \right) = \sum_{i=1}^{k+1} c_i \sigma_i u_i$$

Proof for the Spectral Norm - Part 3: The Lower Bound

- We know $Bz = 0$ and $z = \sum_{i=1}^{k+1} c_i v_i$ for some scalars c_i with $\sum c_i^2 = 1$.
- The squared norm of the error is:

$$\|A - B\|_2^2 \geq \|(A - B)z\|_2^2 = \|Az\|_2^2$$

- We compute Az:

$$Az = \left(\sum_{j=1}^{r} \sigma_j u_j v_j^T \right) \left(\sum_{i=1}^{k+1} c_i v_i \right) = \sum_{i=1}^{k+1} c_i \sigma_i u_i$$

- Taking the squared norm:

$$\|Az\|_2^2 = \left\| \sum_{i=1}^{k+1} c_i \sigma_i u_i \right\|_2^2 = \sum_{i=1}^{k+1} c_i^2 \sigma_i^2$$

Proof for the Spectral Norm - Part 3: The Lower Bound

- Since $\sigma_i \geq \sigma_{k+1}$ for $i \leq k+1$:

$$\sum_{i=1}^{k+1} c_i^2 \sigma_i^2 \geq \sigma_{k+1}^2 \sum_{i=1}^{k+1} c_i^2 = \sigma_{k+1}^2$$

Proof for the Spectral Norm - Part 3: The Lower Bound

- Since $\sigma_i \geq \sigma_{k+1}$ for $i \leq k+1$:

$$\sum_{i=1}^{k+1} c_i^2 \sigma_i^2 \geq \sigma_{k+1}^2 \sum_{i=1}^{k+1} c_i^2 = \sigma_{k+1}^2$$

- So, $\|A - B\|_2^2 \geq \sigma_{k+1}^2$, which completes the proof.