Singular Value Decomposition

Geometric Algorithms
Lecture 26

CAS CS 132

Objectives

1. Introduce the **singular value decomposition** (probably the most important matrix decomposition for computer science)

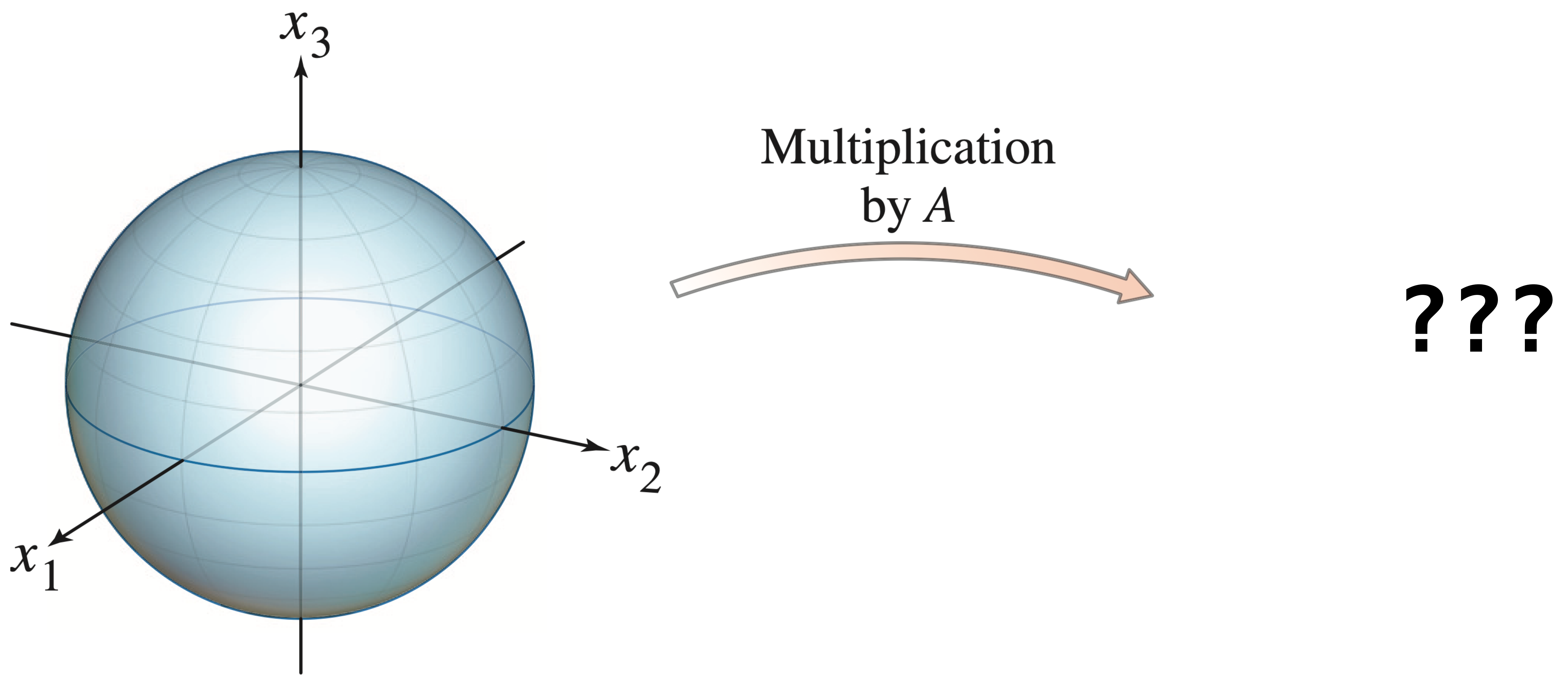
2. Talk very briefly about what to do after this course if you want (or have to) to see more linear algebra

3. Fill out course evals(!)

Motivation

Question

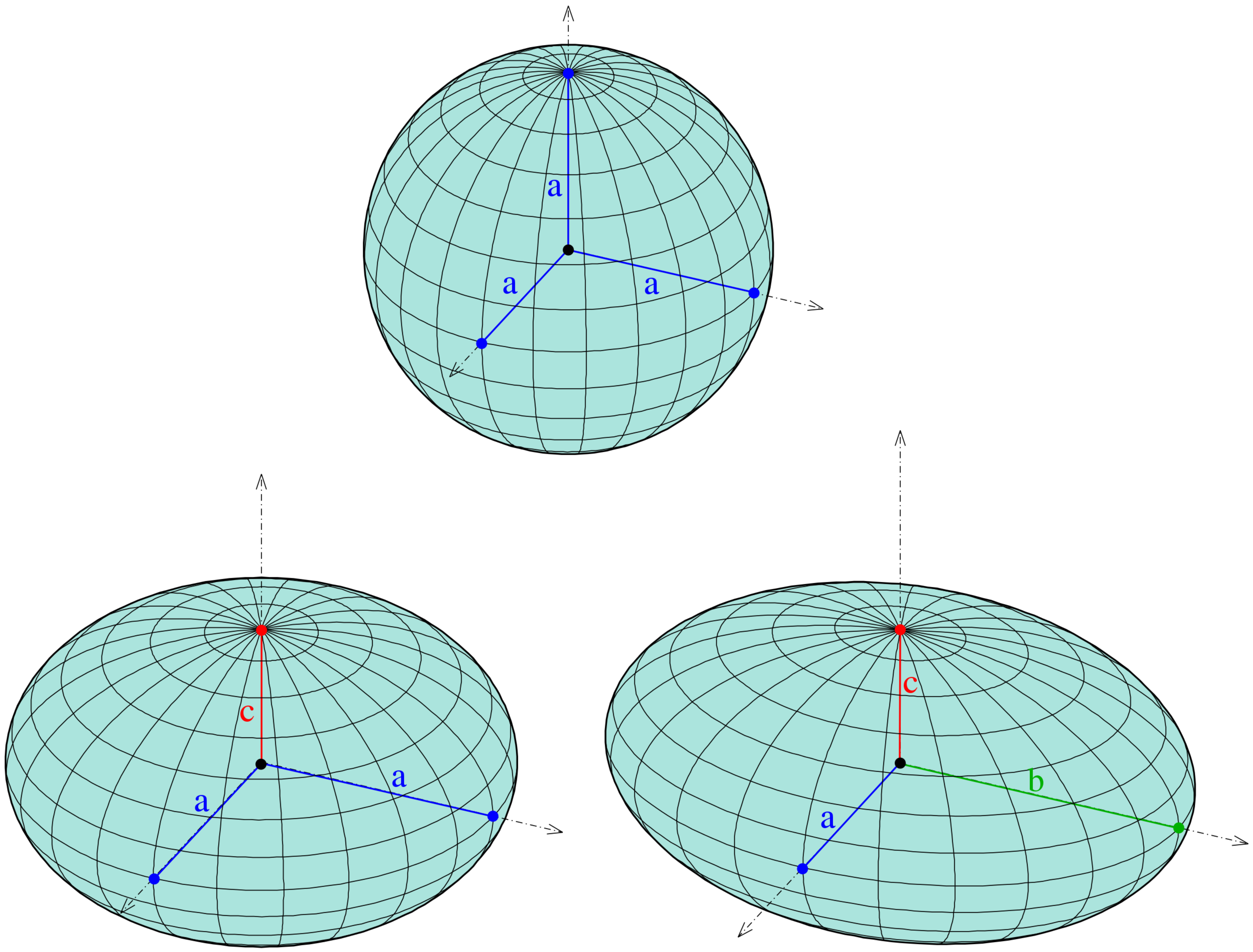
What shape is a the unit sphere after a linear transformation?



Ellipsoids

Ellipsoids are spheres "stretched" in orthogonal directions called the **axes of symmetry** or the **principle axes.**

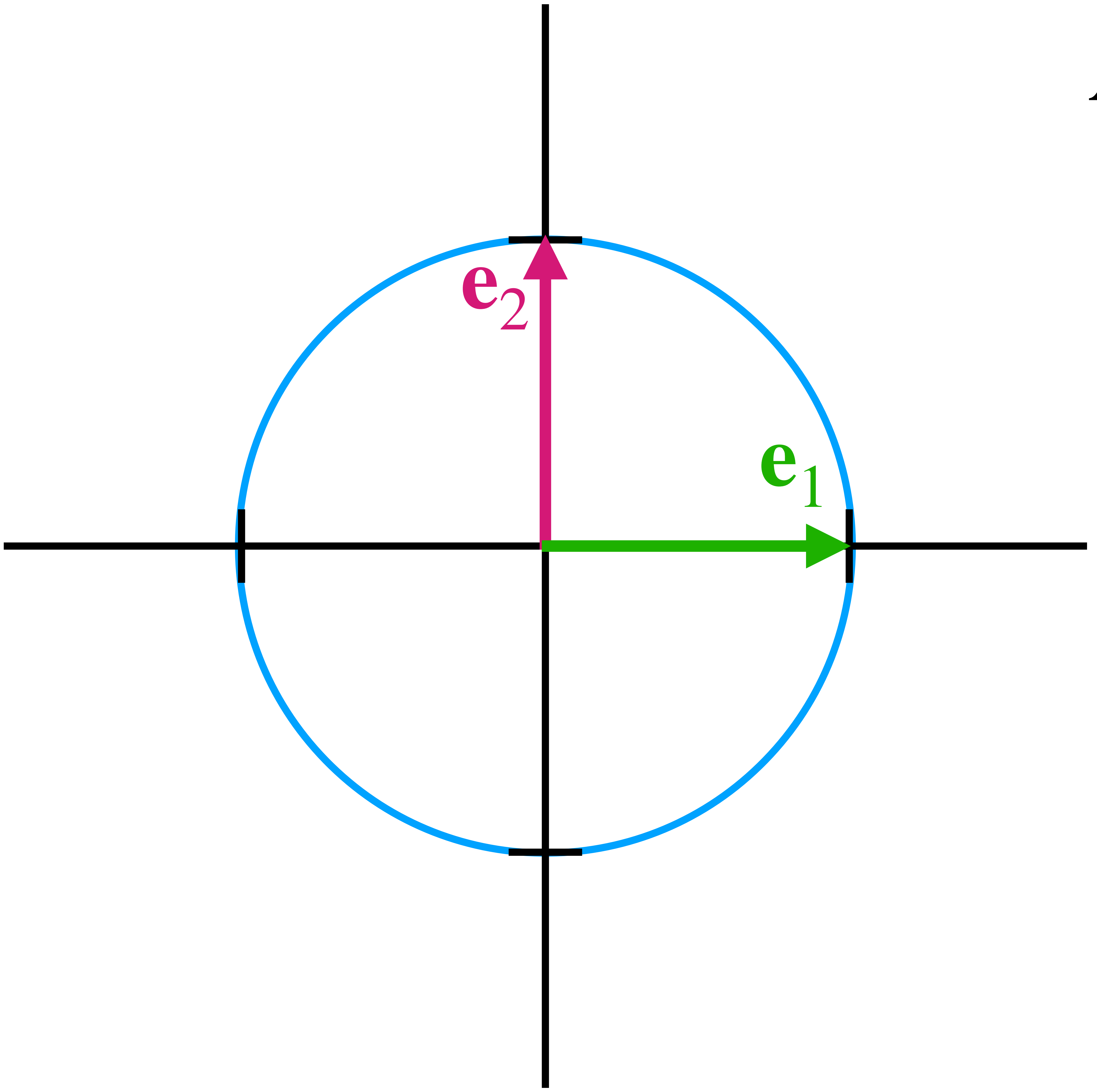
Linear transformations maps <u>spheres</u> to <u>ellipsoids</u>.

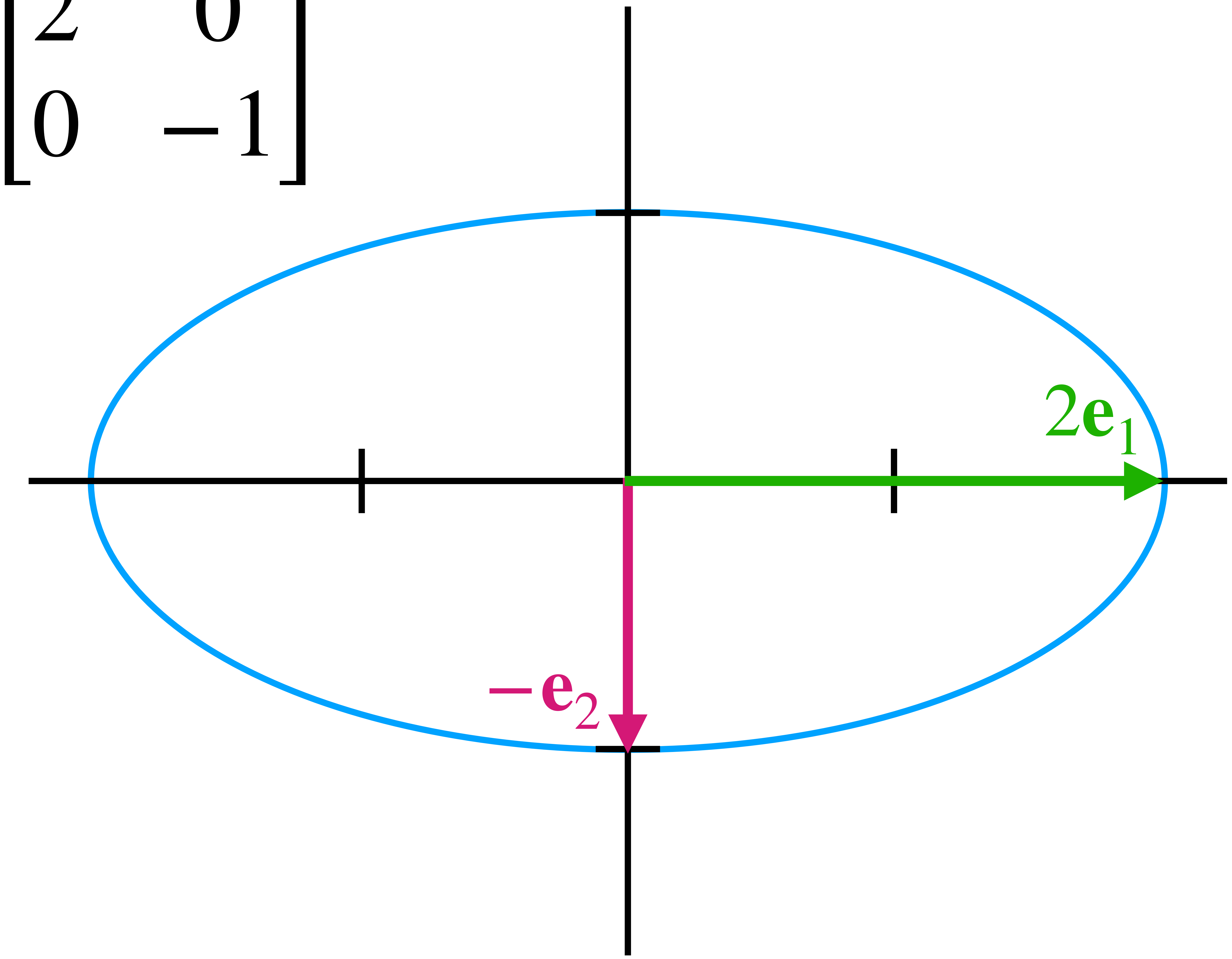


https://commons.wikimedia.org/wiki/File:Ellipsoide.svg

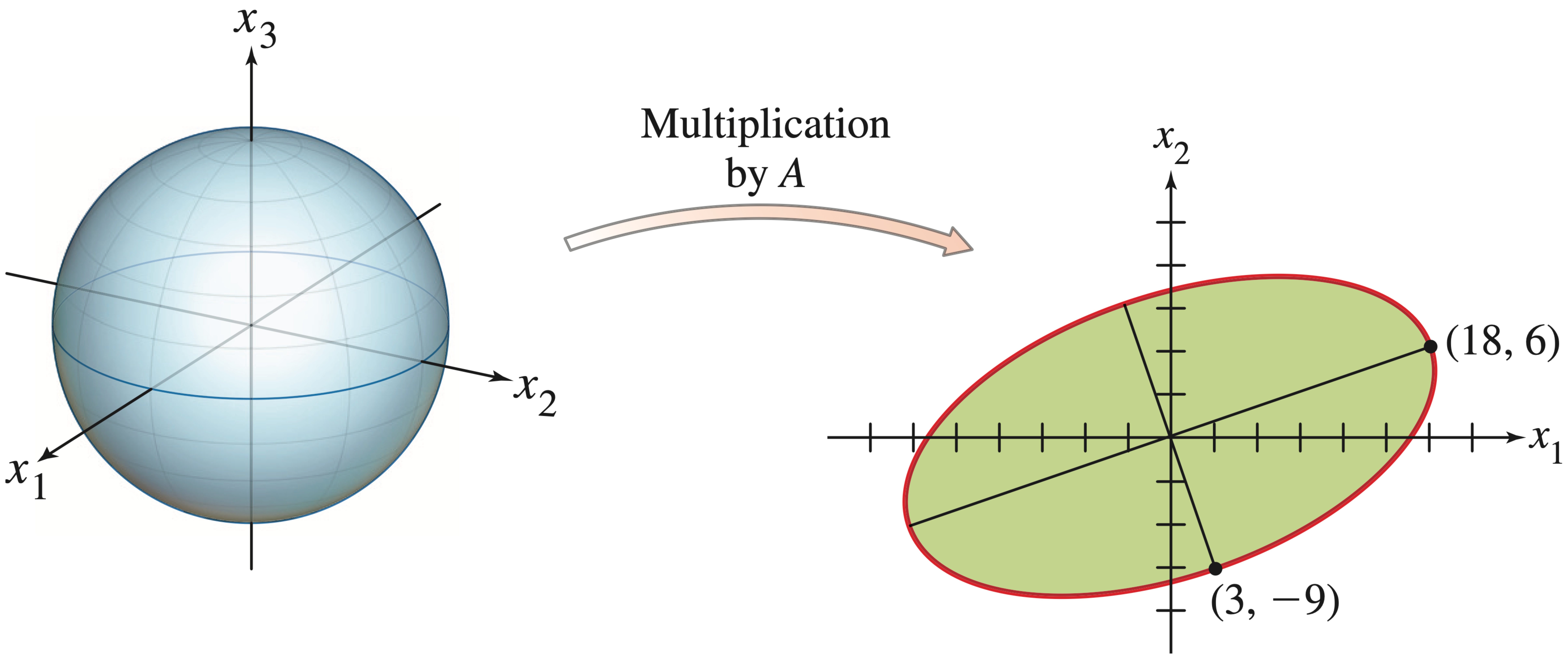
Simple Example : Scaling Matrices

$$A = \begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix}$$

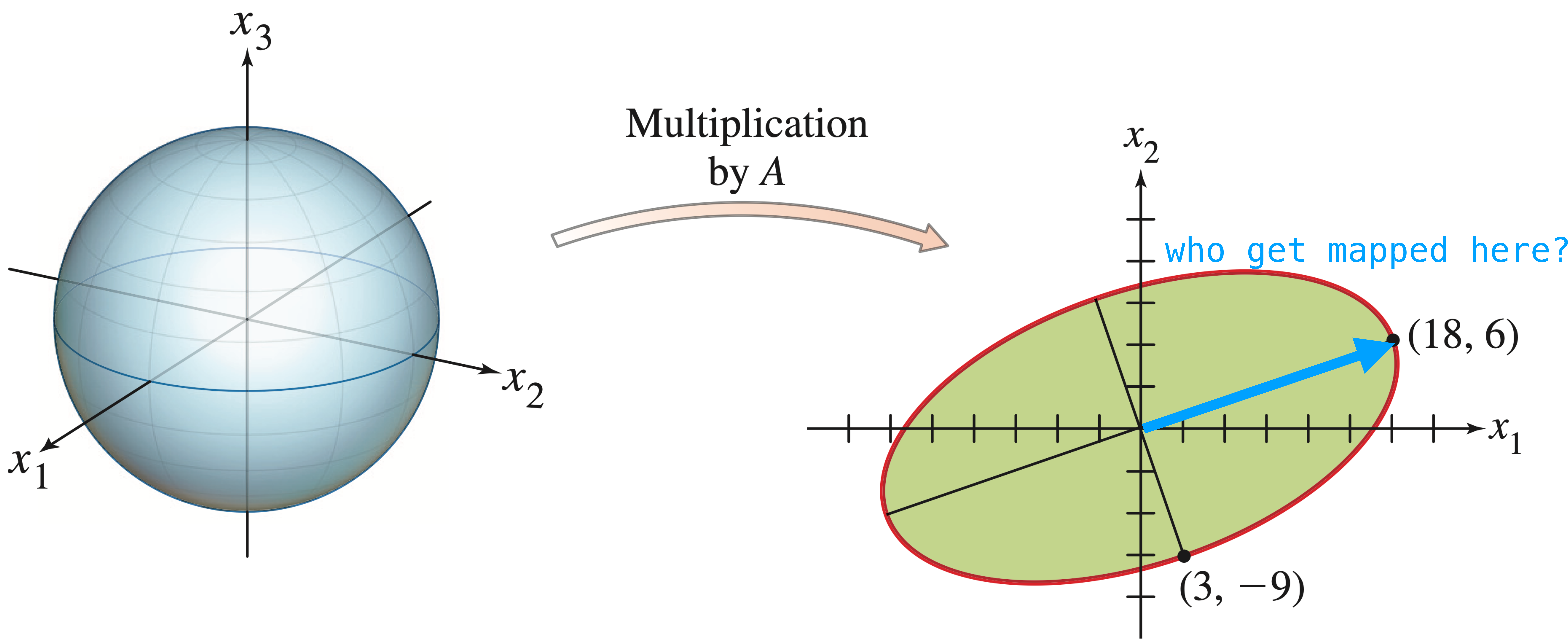




The Picture

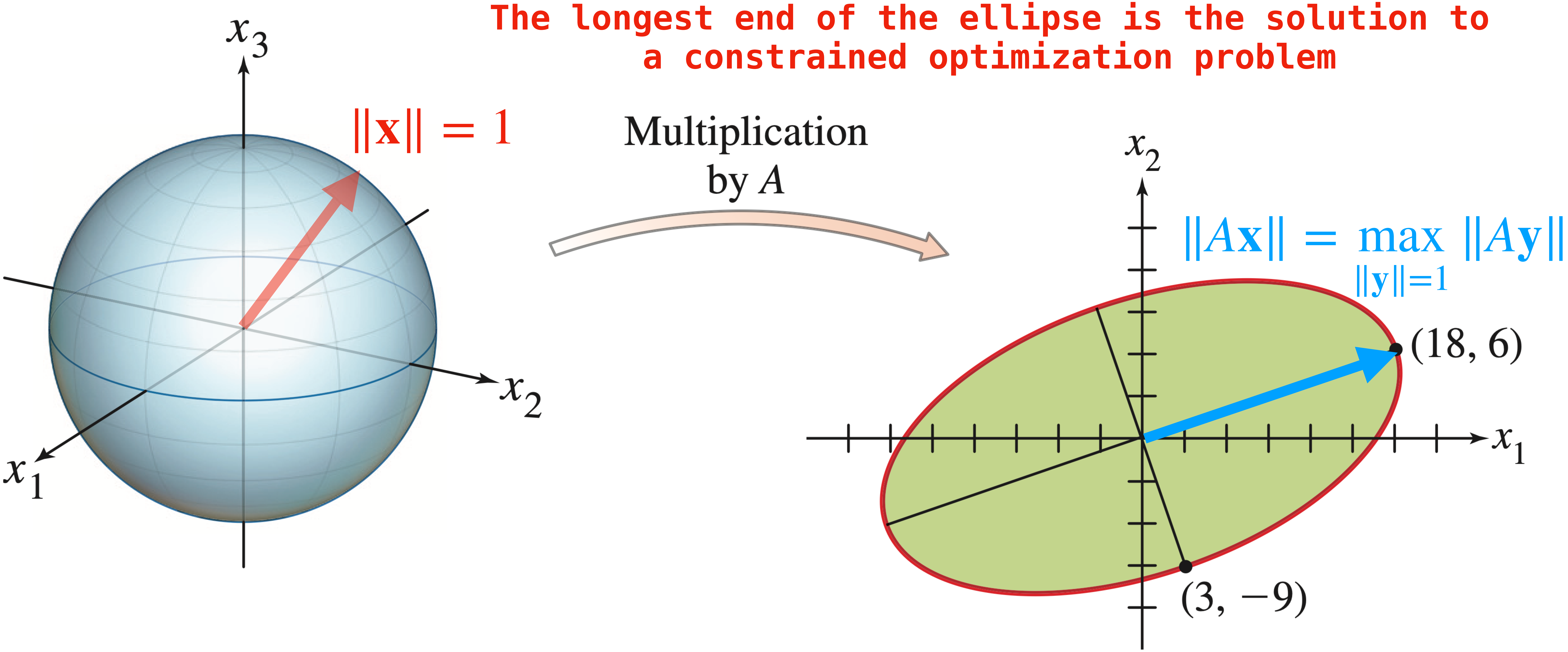


The Picture

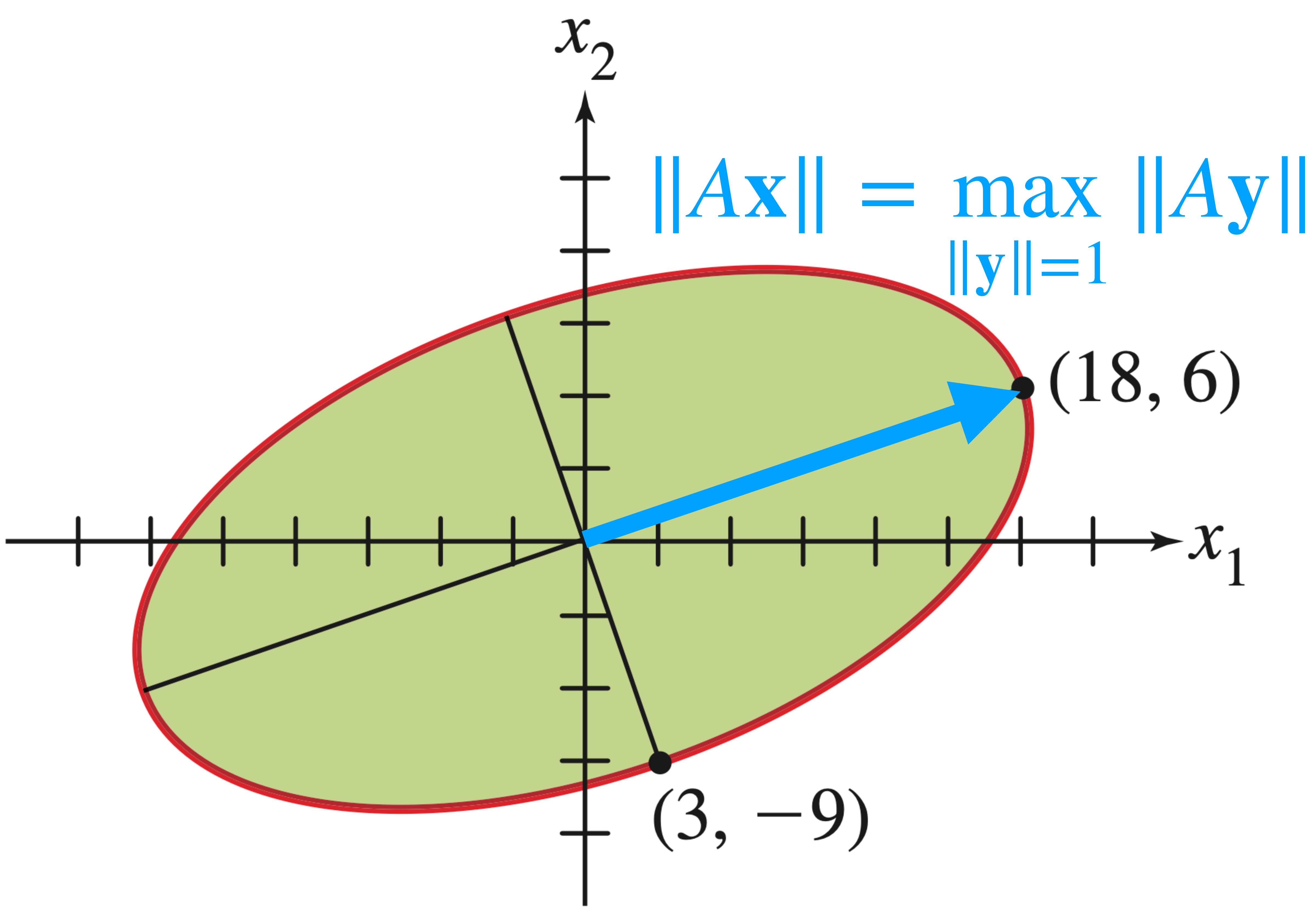


The Picture

The longest end of the ellipse is the solution to a constrained optimization problem

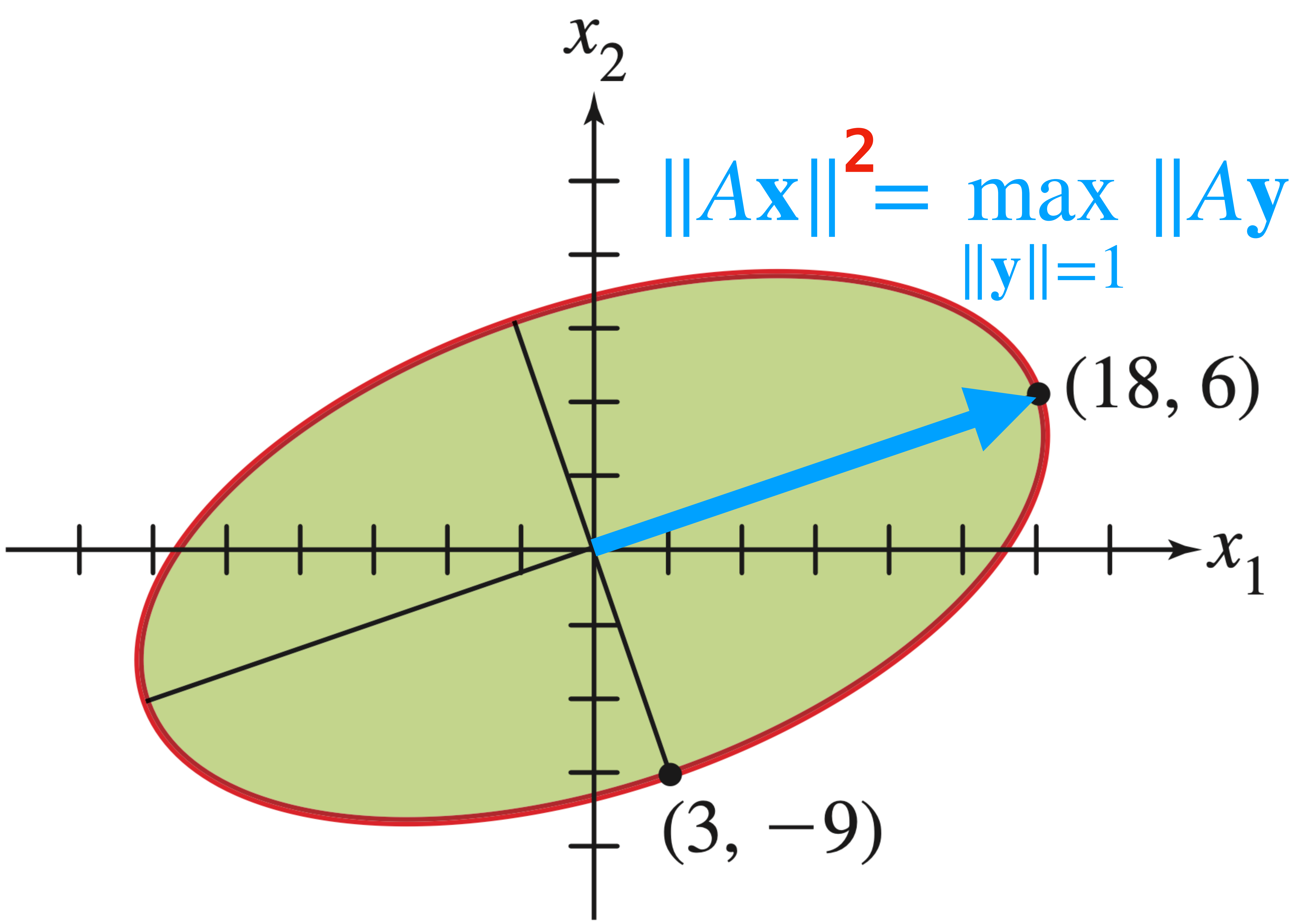


The Picture




```
This is not a quadratic form...
```

The Picture



$$\|A\mathbf{x}\|^2 = \max_{\|\mathbf{y}\|=1} \|A\mathbf{y}\|^2$$

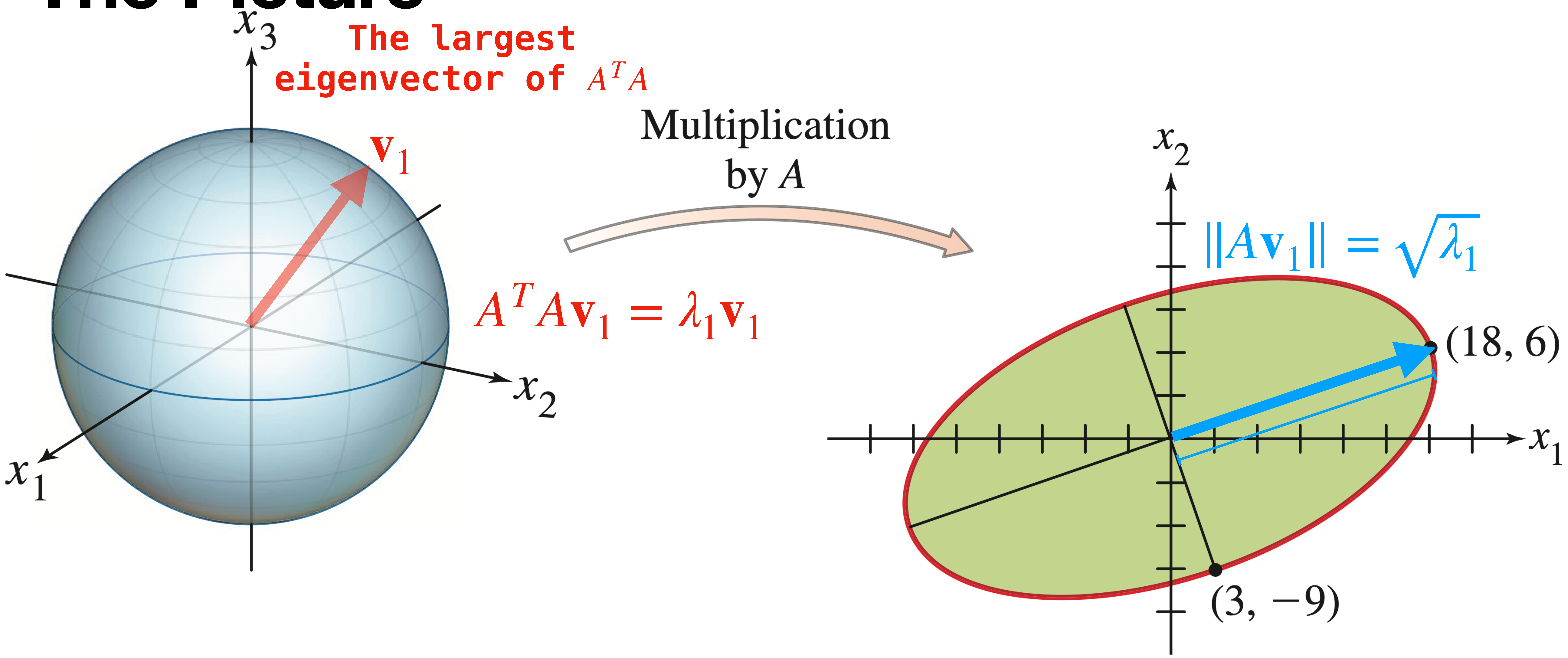
But this is.

This is not a quadratic form...

A Quadratic Form

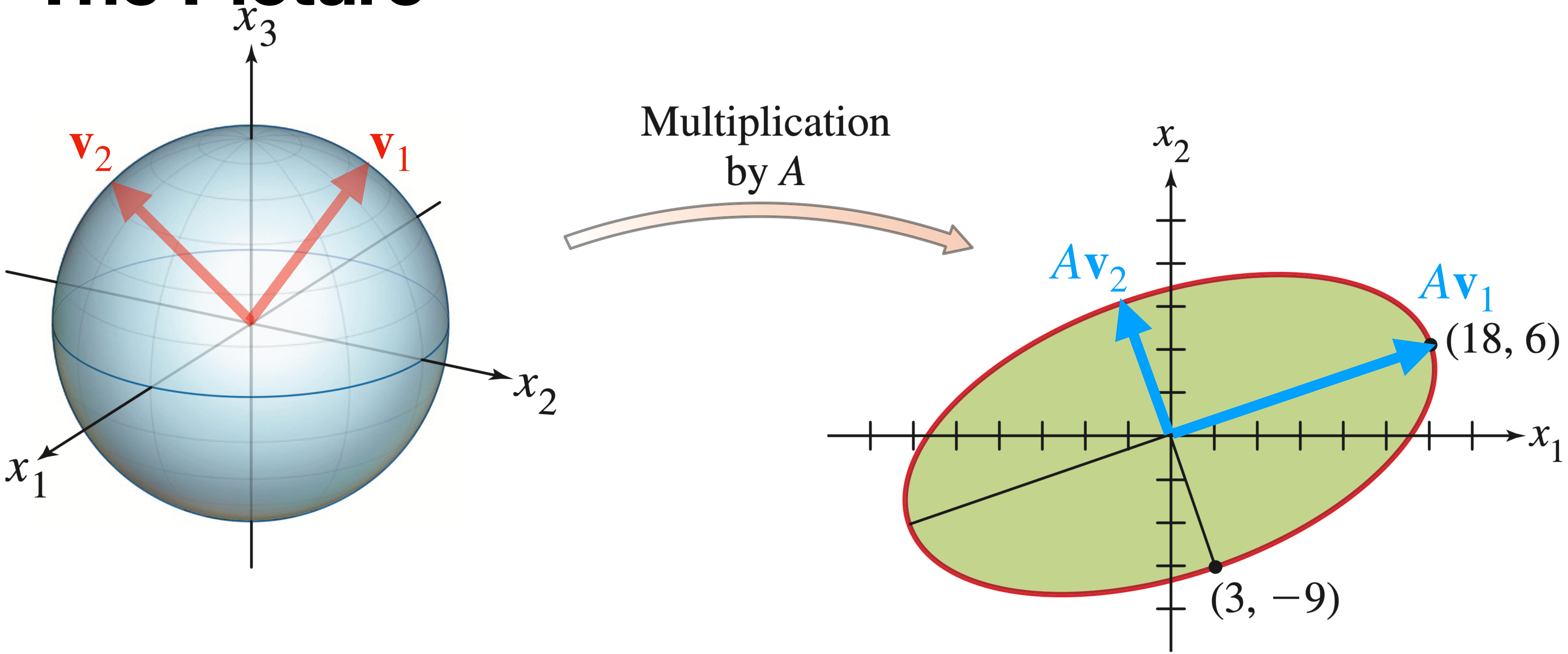
What does $\|A\mathbf{x}\|^2$ look like?:

The Picture



$\mathbf{v}_1$ solves the constrained optimization problem.

The Picture



The second eigenvector is sent to the *minimum* principle axis

Properties of $A^T A$

Properties of $A^T A$

» It's symmetric

Properties of $A^T A$

- It's symmetric
- So its <u>orthogonally diagonalizable</u>

Properties of $A^T A$

- It's symmetric
- So its orthogonally diagonalizable
- **There is an orthogonal basis of eigenvectors**

Properties of $A^T A$

» It's symmetric

» So its orthogonally diagonalizable

» **There is an orthogonal basis of eigenvectors**

» It's eigenvalues are nonnegative

Properties of A^TA

- It's symmetric
- So its orthogonally diagonalizable
- **There is an orthogonal basis of eigenvectors**
- It's eigenvalues are nonnegative
- **It's positive semidefinite**

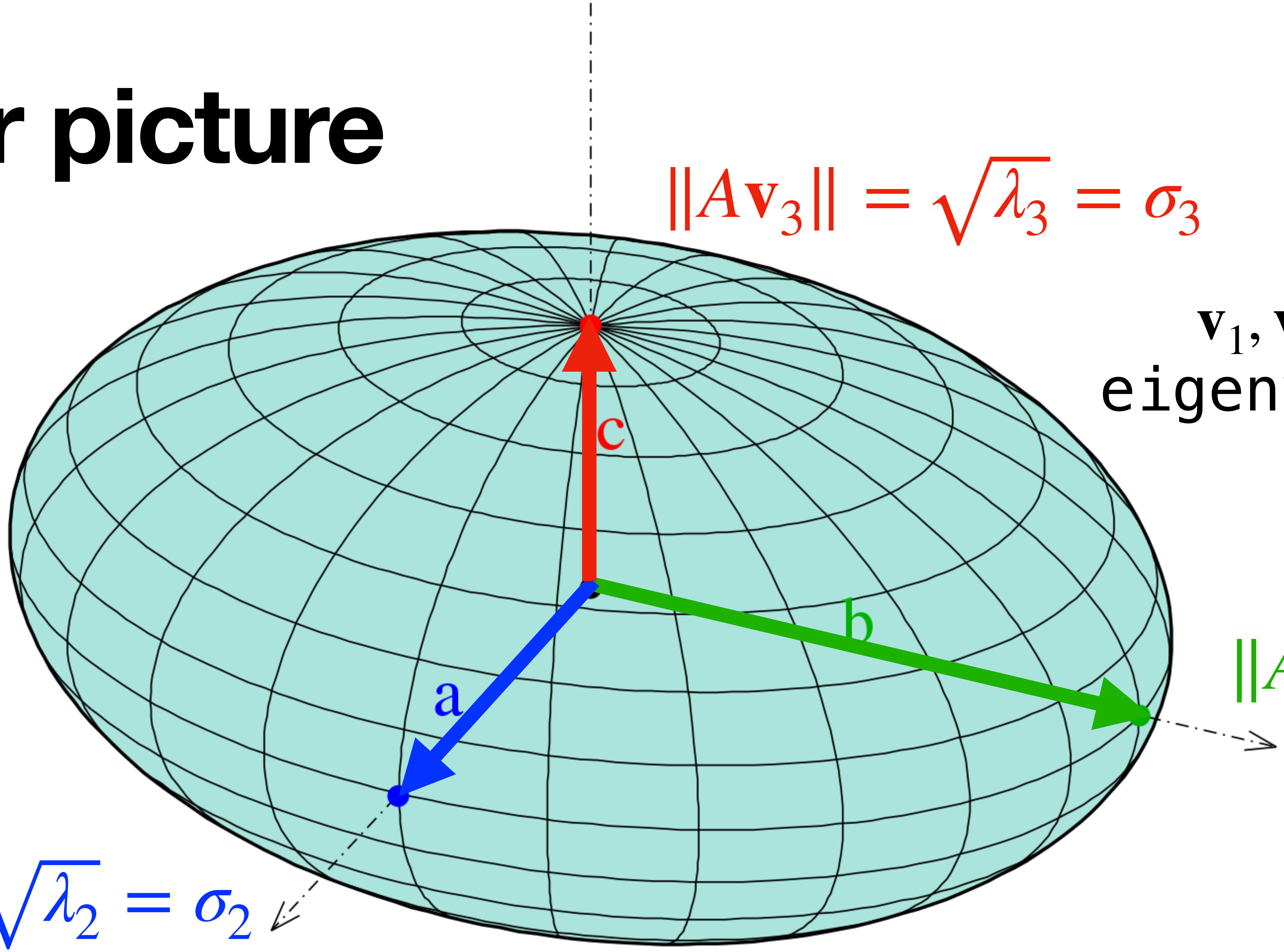
Singular Values

Definition. For an $m \times n$ matrix A, the **singular values** of A are the n values

$$\sigma_1 \geq \sigma_2 \ldots \geq \sigma_n \geq 0$$

where $\sigma_i = \sqrt{\lambda_i}$ and λ_i is an eigenvalue of $A^T A$.

Another picture



$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are the eigenvectors of A^TA

The **singular values** are the lengths of the *axes of symmetry* of the ellipsoid after transforming the unit sphere.

https://commons.wikimedia.org/wiki/File:Ellipsoide.svg

Every $m \times n$ matrix transforms the unit m–sphere into an n–ellipsoid

So <u>every</u> $m \times n$ matrix has n singular values

What else can we say?

Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be an **orthogonal** eigenbasis of $\mathbb{R}^n$ for A^TA and suppose A has r nonzero singular values

Theorem. $A\mathbf{v}_1, \ldots, A\mathbf{v}_r$ is an orthogonal basis of $\text{Col}(A)$

What else can we say?

Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be an **orthogonal** eigenbasis of $\mathbb{R}^n$ for A^TA and suppose A has r nonzero singular values

Theorem. $A\mathbf{v}_1, \ldots, A\mathbf{v}_r$ is an orthogonal basis of Col(A)

This is the most important theorem for SVD

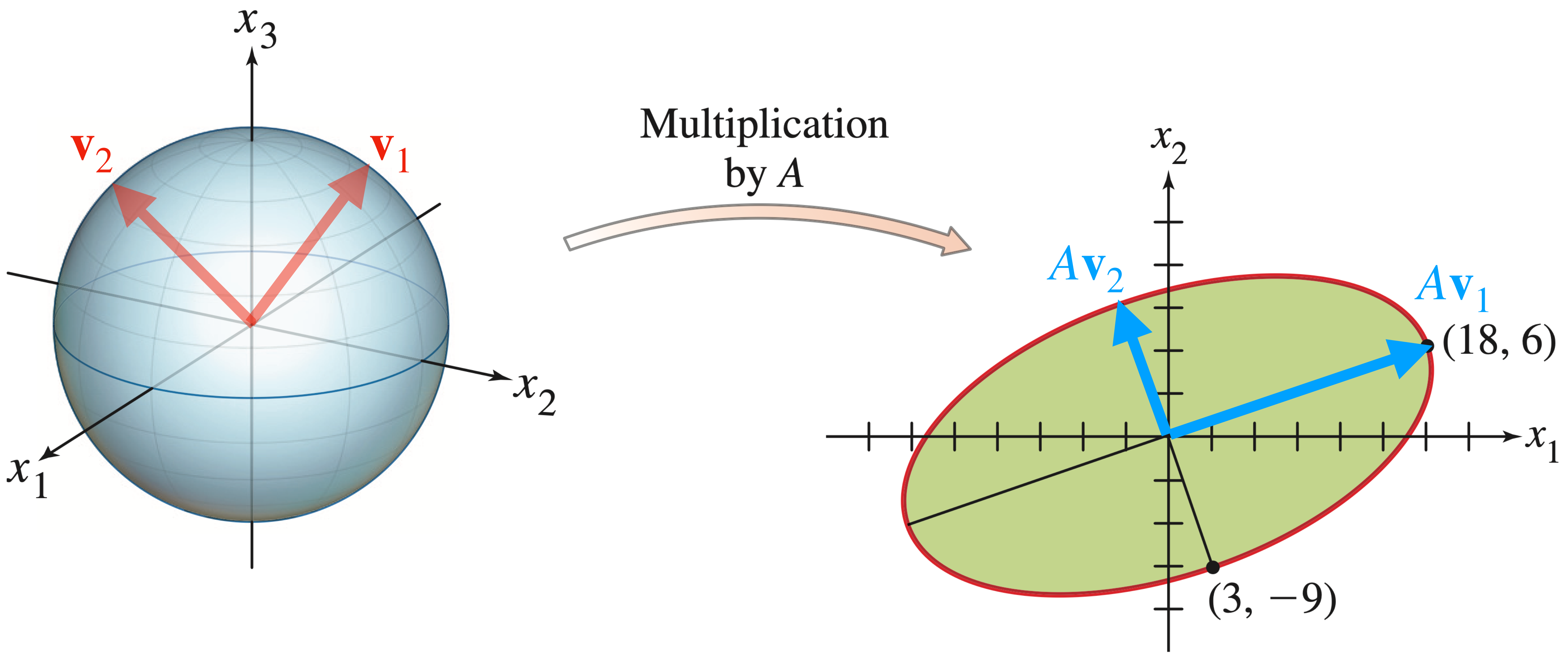
Verifying it

Let's show $A\mathbf{v}_1, \ldots, A\mathbf{v}_r$ are orthogonal (and linearly independent):

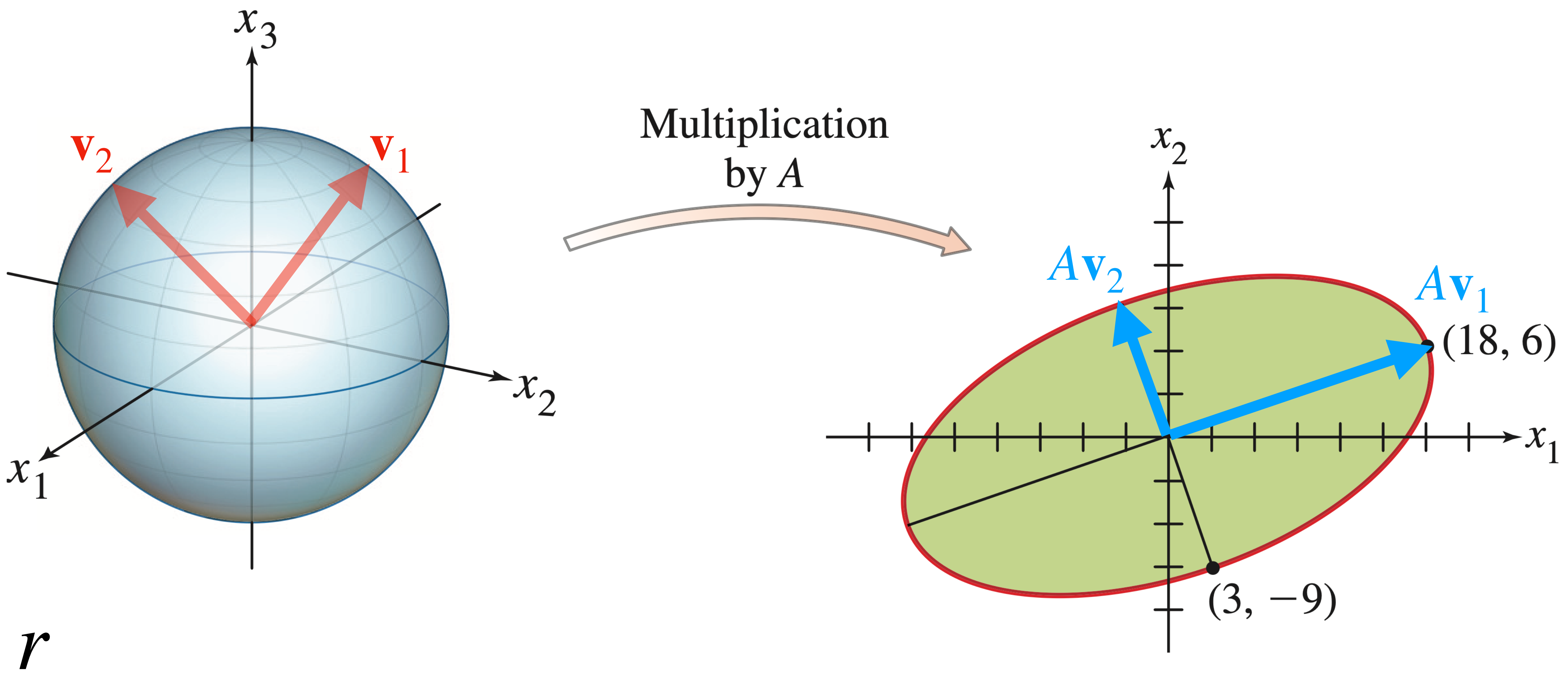
Verifying it

Let's show $A\mathbf{v}_1, \ldots, A\mathbf{v}_r$ span $\text{Col}(A)$:

Putting it all together

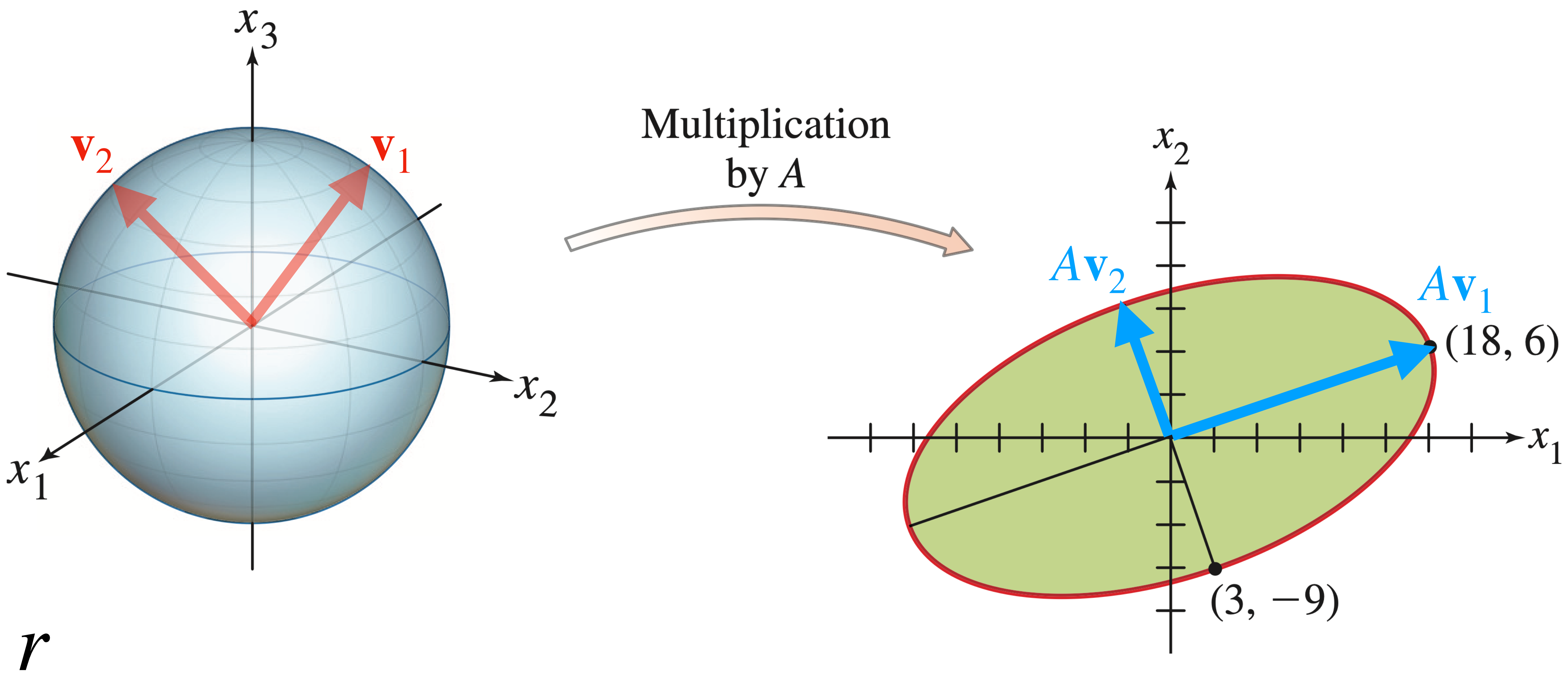


Putting it all together



Let A be an $m \times n$ matrix of rank r

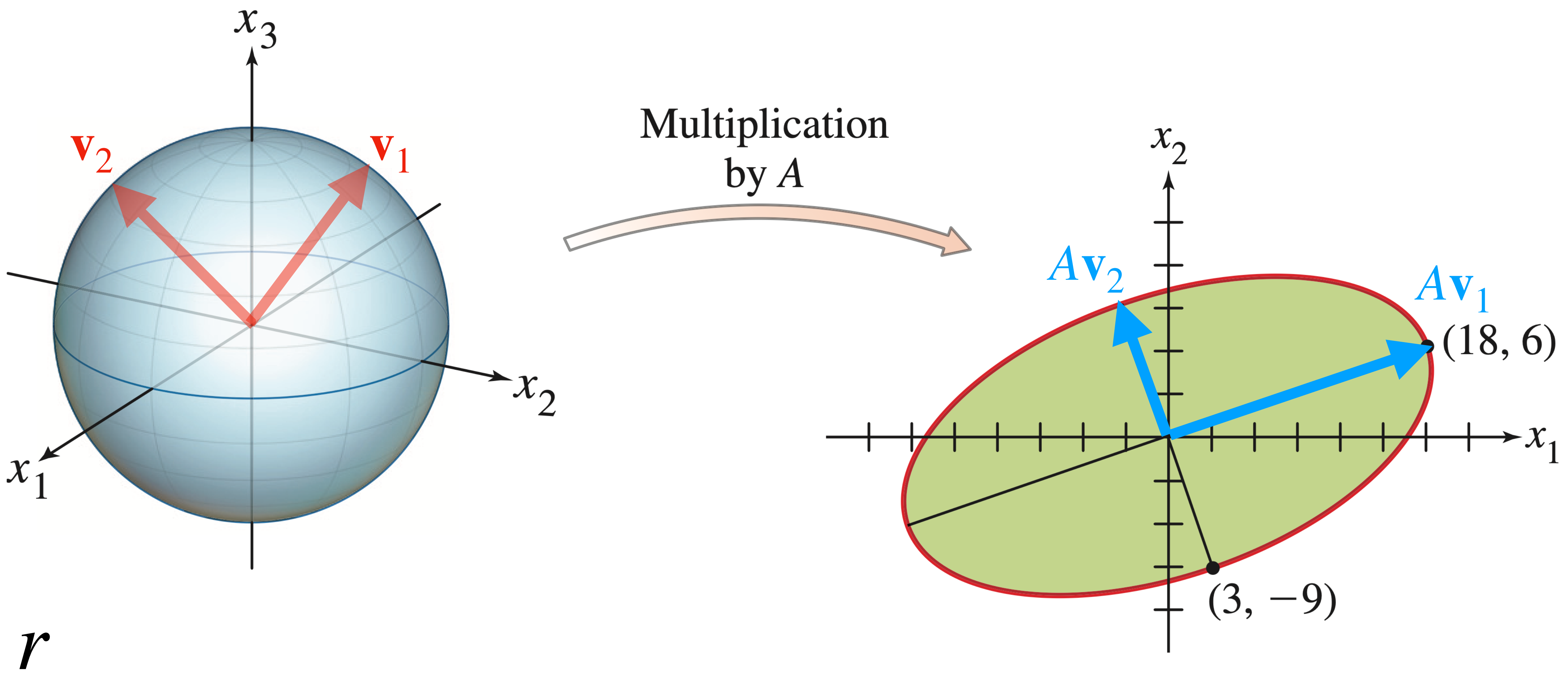
Putting it all together



Let A be an $m \times n$ matrix of rank r

<u>What we know:</u>

Putting it all together

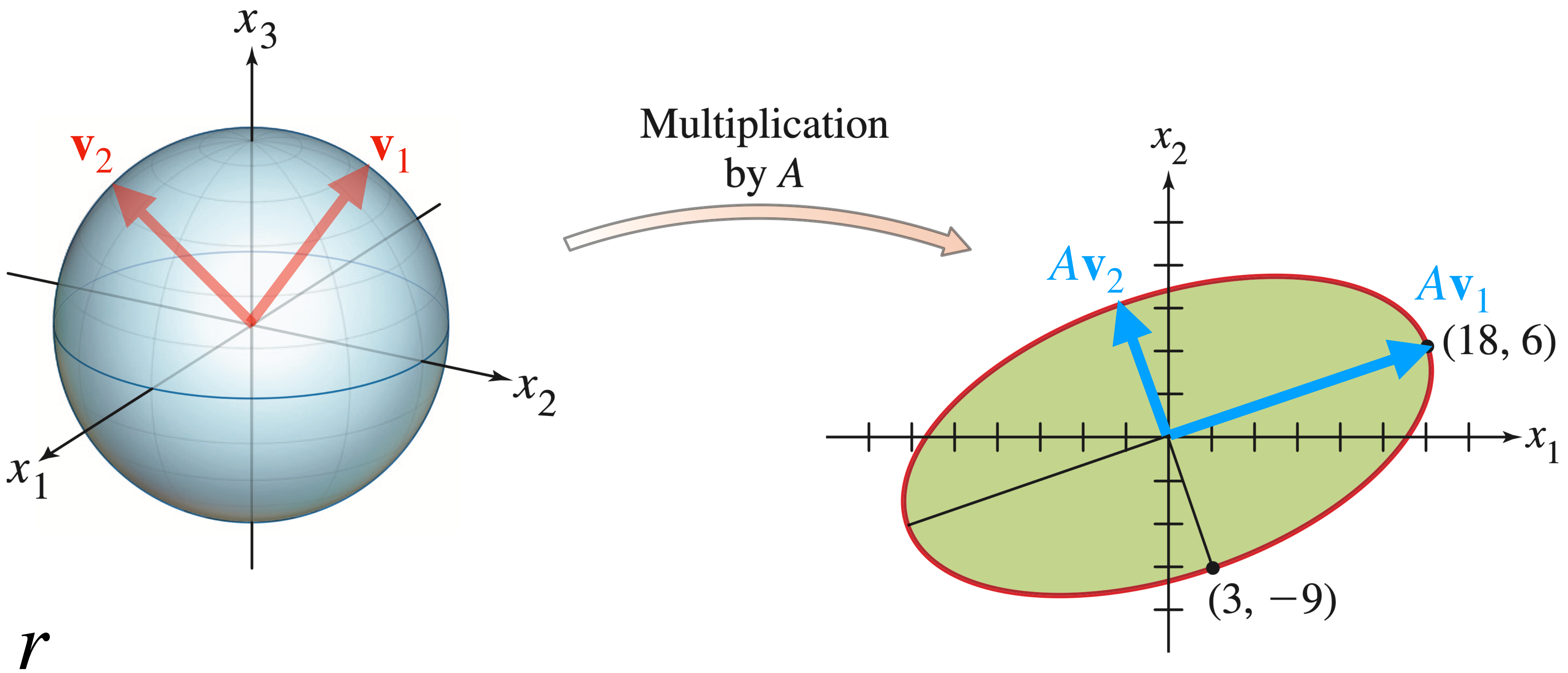


Let A be an $m \times n$ matrix of rank r

What we know:

» We can find orthonormal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ in $\mathbb{R}^n$ such that $A\mathbf{v}_1, \ldots, A\mathbf{v}_r$ in $\mathbb{R}^m$ form an orthogonal basis for $\text{Col}(A)$

Putting it all together



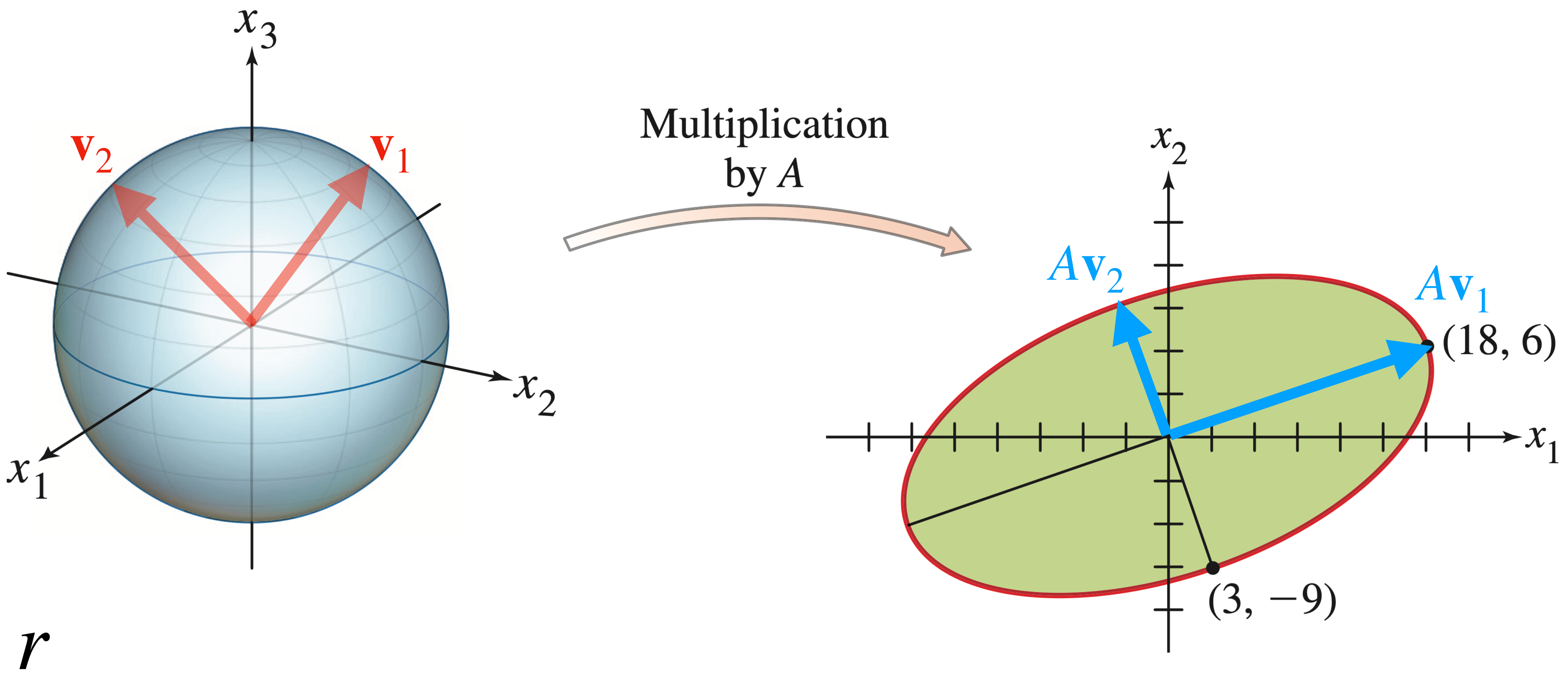
Let A be an $m \times n$ matrix of rank r

<u>What we know:</u>

» We can find orthonormal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ in $\mathbb{R}^n$ such that $A\mathbf{v}_1, \ldots, A\mathbf{v}_r$ in $\mathbb{R}^m$ form an orthogonal basis for $\text{Col}(A)$

» So if we take $\mathbf{u}_i = \dfrac{A\mathbf{v}_i}{\|A\mathbf{v}_i\|}$, we get an **orthonormal** basis of $\text{Col}(A)$

Putting it all together



Let A be an $m \times n$ matrix of rank r

<u>What we know:</u>

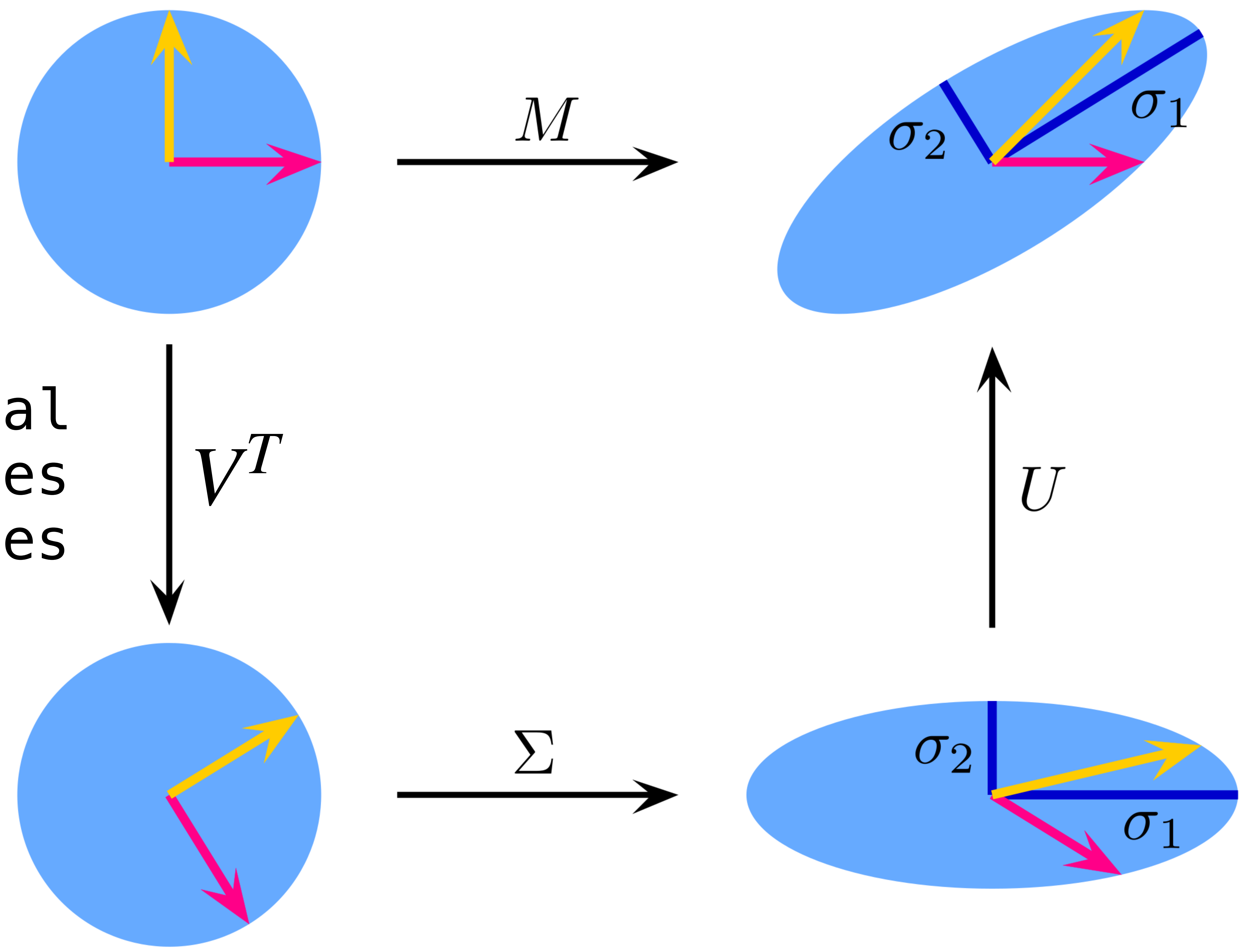
» We can find orthonormal vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ in $\mathbb{R}^n$ such that $A\mathbf{v}_1, \ldots, A\mathbf{v}_r$ in $\mathbb{R}^m$ form an orthogonal basis for $\text{Col}(A)$

» So if we take $\mathbf{u}_i = \dfrac{A\mathbf{v}_i}{\|A\mathbf{v}_i\|}$, we get an **orthonormal** basis of $\text{Col}(A)$

» And we can extend this to $\mathbf{u}_1, \ldots, \mathbf{u}_m$ an orthonormal basis of $\mathbb{R}^m$ (via Gram-Schmidt).

Singular Value Decomposition

High Level View of the Decomposition



<u>recall</u>: Orthogonal matrices preserves lengths and angles

$$M = U \cdot \Sigma \cdot V^T$$

https://commons.wikimedia.org/wiki/File:Singular-Value-Decomposition.svg

The Important Equality

$$\mathbf{u}_i = \frac{A\mathbf{v}_i}{\|A\mathbf{v}_i\|}$$

$$A\mathbf{v}_i = \|A\mathbf{v}_i\|\mathbf{u}_i = \sigma_i\mathbf{u}_i$$

The Important Equality

$$\mathbf{u}_i = \frac{A\mathbf{v}_i}{\|A\mathbf{v}_i\|}$$

$$A\mathbf{v}_i = \|A\mathbf{v}_i\|\mathbf{u}_i = \sigma_i\mathbf{u}_i$$

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$

The Important Equality

$$\mathbf{u}_i = \frac{A\mathbf{v}_i}{\|A\mathbf{v}_i\|}$$

$$A\mathbf{v}_i = \|A\mathbf{v}_i\|\mathbf{u}_i = \sigma_i \mathbf{u}_i$$

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$

What happens when we write this in matrix form?

The Important Equality

$$A[\mathbf{v}_1 \ \ \dots \ \ \mathbf{v}_n] = [\sigma_1 \mathbf{u}_1 \ \ \dots \sigma_n \mathbf{u}_n]$$

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$.

The Important Equality

$$A[\mathbf{v}_1 \ \ \dots \ \ \mathbf{v}_n] = [\sigma_1\mathbf{u}_1 \ \ \dots\sigma_n\mathbf{u}_n]$$

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$.

Let's take $V = [\mathbf{v}_1 \ \ \dots \ \ \mathbf{v}_n]$ and $U = [\mathbf{u}_1 \ \ \dots \ \ \mathbf{u}_m]$ and

The Important Equality

$$A[\mathbf{v}_1 \ \dots \ \mathbf{v}_n] = [\sigma_1\mathbf{u}_1 \ \dots\sigma_n\mathbf{u}_n]$$

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$.

Let's take $V = [\mathbf{v}_1 \ \dots \ \mathbf{v}_n]$ and $U = [\mathbf{u}_1 \ \dots \ \mathbf{u}_m]$ and

$m > n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

or

$m < n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_m & 0 & \dots & 0 \end{bmatrix}$$

or

$m = n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \end{bmatrix}$$

The Important Equality

$$A[\mathbf{v}_1 \ \ldots \ \mathbf{v}_n] = [\sigma_1\mathbf{u}_1 \ \ldots \sigma_n\mathbf{u}_n]$$

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$.

Let's take $V = [\mathbf{v}_1 \ \ldots \ \mathbf{v}_n]$ and $U = [\mathbf{u}_1 \ \ldots \ \mathbf{u}_m]$ and

remember: U is orthonormal

$m > n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_n \\ 0 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 0 \end{bmatrix}$$

or

$m < n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_m & 0 & \ldots & 0 \end{bmatrix}$$

or

$m = n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_n \end{bmatrix}$$

The Important Equality

$$AV = U\Sigma$$

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$.

Let's take $V = [\mathbf{v}_1 \ \dots \ \mathbf{v}_n]$ and $U = [\mathbf{u}_1 \ \dots \ \mathbf{u}_m]$ and

remember: U is orthonormal

$m > n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

or

$m < n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_m & 0 & \dots & 0 \end{bmatrix}$$

or

$m = n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \end{bmatrix}$$

The Important Equality

$$\underset{m \times n}{A}\underset{n \times n}{V} = \underset{m \times m}{U}\underset{m \times n}{\Sigma}$$

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$.

Let's take $V = [\mathbf{v}_1 \ \dots \ \mathbf{v}_n]$ and $U = [\mathbf{u}_1 \ \dots \ \mathbf{u}_m]$ and

remember: U is orthonormal

$m > n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

or

$m < n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_m & 0 & \dots & 0 \end{bmatrix}$$

or

$m = n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \end{bmatrix}$$

The Important Equality

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

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$.

Let's take $V = [\mathbf{v}_1 \ \dots \ \mathbf{v}_n]$ and $U = [\mathbf{u}_1 \ \dots \ \mathbf{u}_m]$ and

remember: U is orthonormal

$m > n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

or

$m < n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_m & 0 & \dots & 0 \end{bmatrix}$$

or

$m = n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \end{bmatrix}$$

The Important Equality

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

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$.

Let's take $V = [\mathbf{v}_1 \ \dots \ \mathbf{v}_n]$ and $U = [\mathbf{u}_1 \ \dots \ \mathbf{u}_m]$ and

remember: U is orthonormal

$m > n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

or

$m < n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_m & 0 & \dots & 0 \end{bmatrix}$$

or

$m = n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \end{bmatrix}$$

The Important Equality

singular value decomposition

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

Remember that $\sigma_i = \sqrt{\lambda_i}$ is the singular value, which is the length $\|A\mathbf{v}_i\|$.

Let's take $V = [\mathbf{v}_1 \ \dots \ \mathbf{v}_n]$ and $U = [\mathbf{u}_1 \ \dots \ \mathbf{u}_m]$ and

remember: U is orthonormal

$m > n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

or

$m < n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_m & 0 & \dots & 0 \end{bmatrix}$$

or

$m = n$

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \end{bmatrix}$$

Singular Value Decomposition

Theorem. For a $m \times n$ matrix A, there are *orthogonal* matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ such that

$$A = \underbrace{U}_{m \times m} \underbrace{\Sigma}_{m \times n} \underbrace{V^T}_{n \times n}$$

where diagonal entries* of Σ are $\sigma_1, \ldots, \sigma_n$ the singular values of A.

* these are diagonal entries in a non-square matrix.

Singular Value Decomposition

Theorem. For a $m \times n$ matrix A, there are *orthogonal* matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ such that

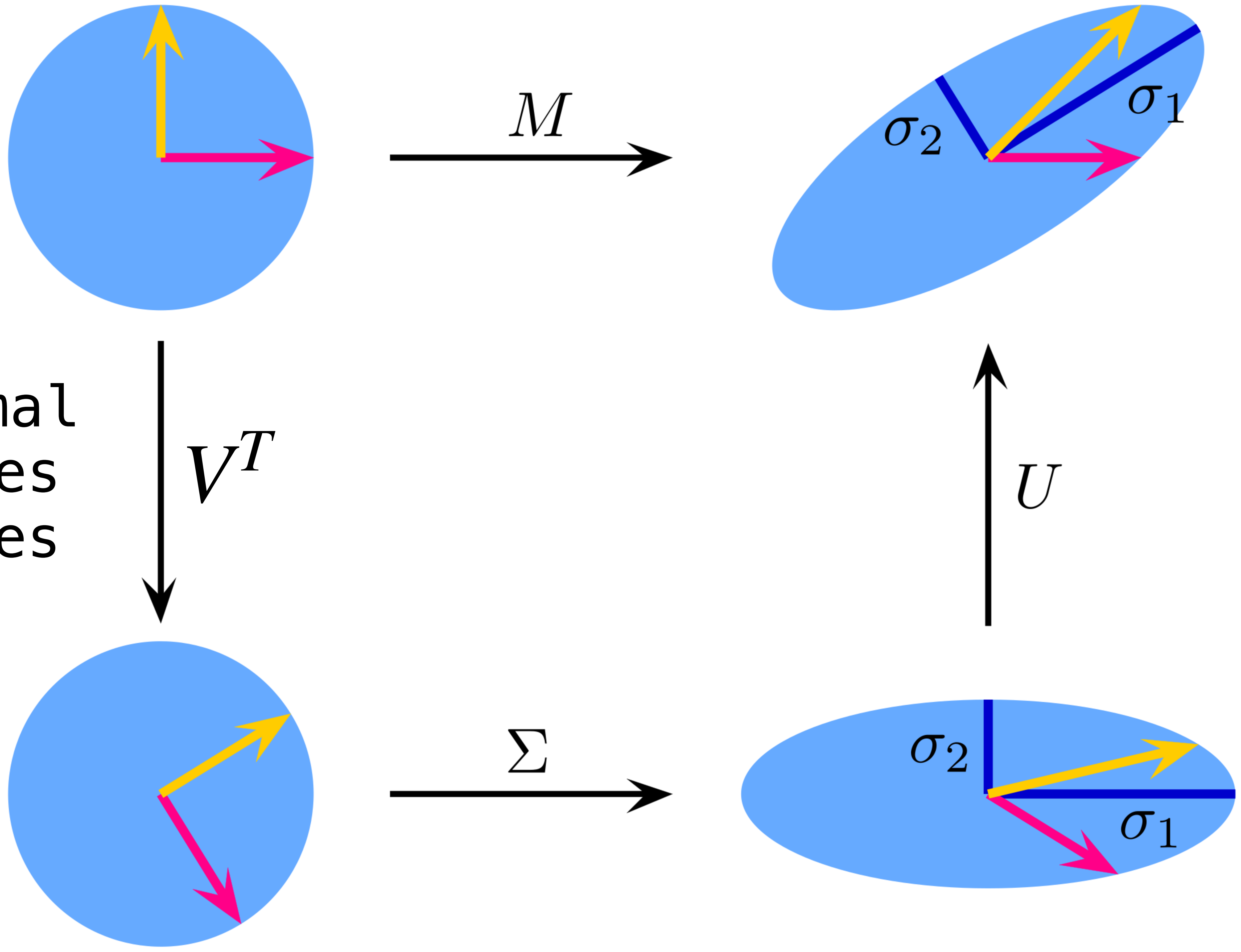
left singular vectors **right singular vectors**

$$A = \underbrace{U}_{m \times m} \underbrace{\Sigma}_{m \times n} \underbrace{V^T}_{n \times n}$$

where diagonal entries* of Σ are $\sigma_1, \ldots, \sigma_n$ the singular values of A.

* these are diagonal entries in a non-square matrix.

The Picture (Again)



recall: Orthonormal matrices preserves lengths and angles

$$M = U \cdot \Sigma \cdot V^T$$

https://commons.wikimedia.org/wiki/File:Singular-Value-Decomposition.svg

How To: Finding a SVD

Step 1: Set up Σ

$$\begin{bmatrix} 1 & -1 \\ -2 & 2 \\ 2 & -2 \end{bmatrix}$$

The **singular values** are the square roots of the eigenvalues of A^TA (or AA^T):

Step 2: Set up V

$$\begin{bmatrix} 1 & -1 \\ -2 & 2 \\ 2 & -2 \end{bmatrix}$$

Find an orthonormal eigenbasis for A^TA:

Step 3: Set up U (Part 1)

$$\begin{bmatrix} 1 & -1 \\ -2 & 2 \\ 2 & -2 \end{bmatrix}$$

If $\mathbf{v}_1 \ldots, \mathbf{v}_n$ is an eigenbasis of $\mathbb{R}^n$ (in decreasing order of eigenvalue), then $A\mathbf{v}_1, \ldots, A\mathbf{v}_r$ is an eigenbasis of $\text{Col}(A)$ (where r is the rank of A). These vectors can be normalized and made the first r columns of U:

Step 4: Set up U (Part 2)

$$\begin{bmatrix} 1 & -1 \\ -2 & 2 \\ 2 & -2 \end{bmatrix}$$

If $m > r$, then extend $\mathbf{u}_1, \ldots, \mathbf{u}_r$ until it has m orthonormal vectors:

Step 5: Put everything together

$$\begin{bmatrix} 1 & -1 \\ -2 & 2 \\ 2 & -2 \end{bmatrix}$$

SVD in NumPy

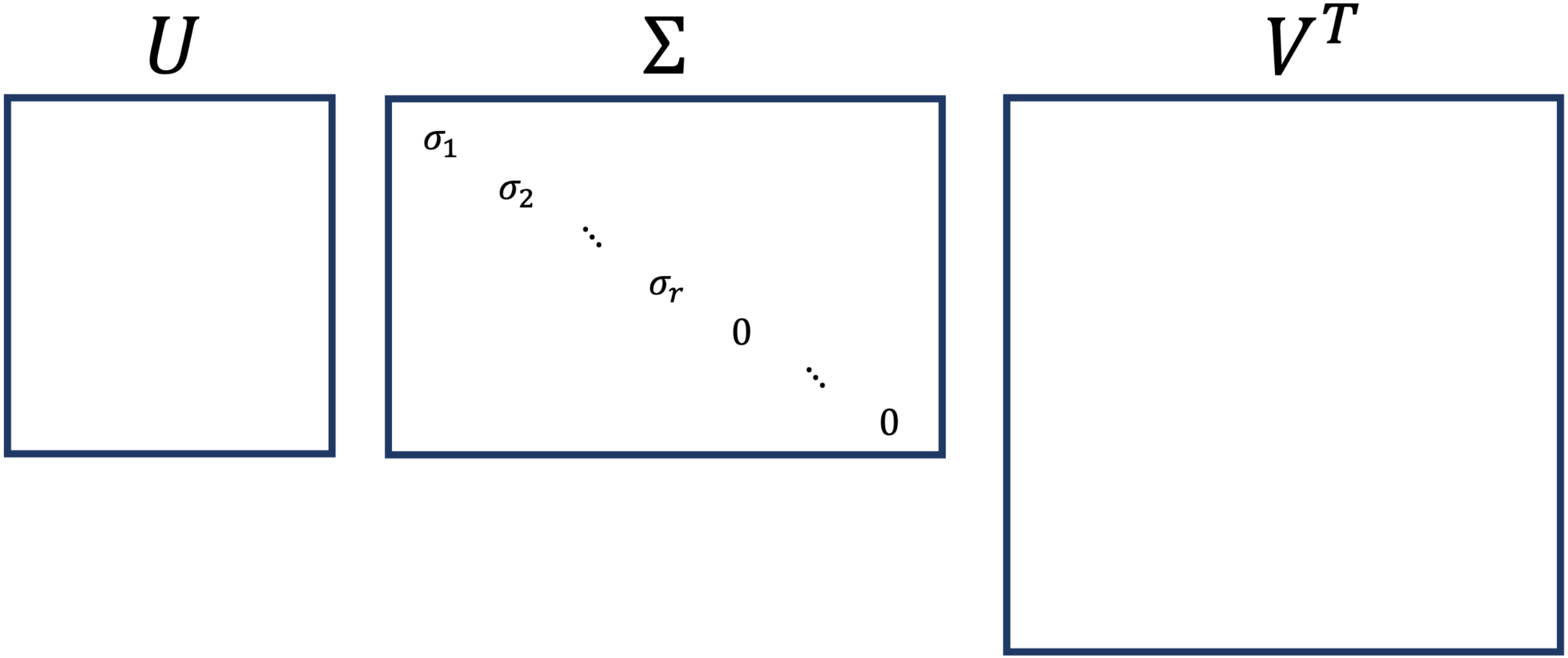
In reality, we will almost never build SVDs by hand. We can use:

numpy.linalg.svd

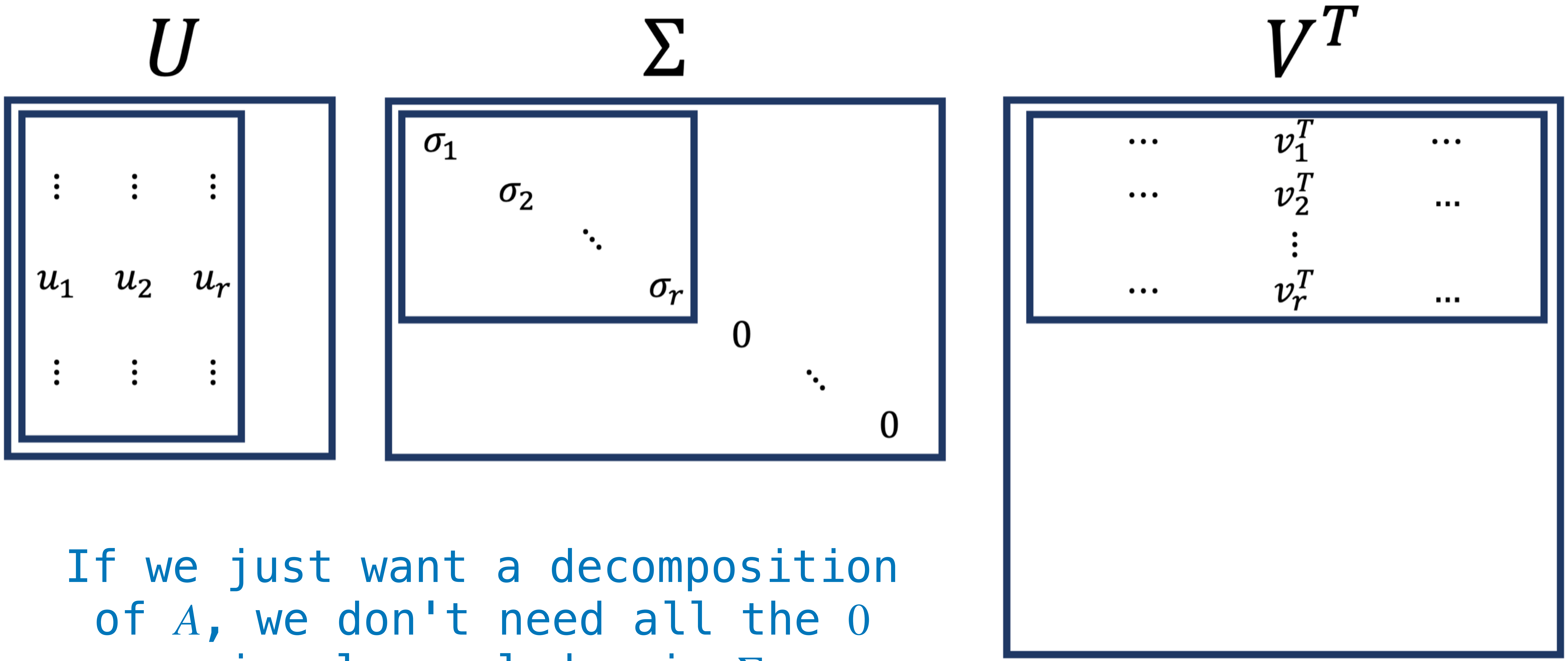
Let's do a quick demo...

Pseudoinverses

SVD (The Picture)



Reduced SVD (The Picture)



If we just want a decomposition of A, we don't need all the 0 singular valudes in Σ

The Reduced SVD

Theorem. For every matrix A of rank r, there is an orthonormal matrix $U \in \mathbb{R}^{m \times r}$, a diagonal matrix $\Sigma \in \mathbb{R}^{r \times r}$ with **positive** entries on the diagonal, and an orthonormal matrix $V \in \mathbb{R}^{n \times r}$ such that

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

The Pseudoinverse

Definition. Given a reduced SVD $A = U\Sigma V^T$, the ***pseudoinverse*** of A is $A^+ = V\Sigma^{-1}U^T$

Theorem. $A^+\mathbf{b}$ is the *minimum length* **least squares solution** of $A\mathbf{x} = \mathbf{b}$

(in Python we have ***numpy.linalg.pinv****)*

Recall: Least Squares in NumPy

numpy.linalg.lstsq

`linalg.lstsq(a, b, rcond='warn')` [source]

Return the least-squares solution to a linear matrix equation.

Computes the vector *x* that approximately solves the equation `a @ x = b`. The equation may be under-, well-, or over-determined (i.e., the number of linearly independent rows of *a* can be less than, equal to, or greater than its number of linearly independent columns). If *a* is square and of full rank, then *x* (but for round-off error) is the "exact" solution of the equation. Else, *x* minimizes the Euclidean 2-norm $||b - ax||$. If there are multiple minimizing solutions, the one with the smallest 2-norm $||x||$ is returned.

Parameters:

a : *(M, N) array_like*

"Coefficient" matrix.

b : *{(M,), (M, K)} array_like*

Ordinate or "dependent variable" values. If *b* is two-dimensional, the least-squares solution is calculated for each of the *K* columns of *b*.

rcond : *float, optional*

Recall: Least Squares in NumPy

numpy.linalg.lstsq

`linalg.lstsq(a, b, rcond='warn')` [source]

Return the least-squares solution to a linear matrix equation.

Computes the vector *x* that approximately solves the equation `a @ x = b`. The equation may be under-, well-, or over-determined (i.e., the number of linearly independent rows of *a* can be less than, equal to, or greater than its number of linearly independent columns). If *a* is square and of full rank, then *x* (but for round-off error) is the "exact" solution of the equation. Else, *x* minimizes the Euclidean 2-norm $||b - ax||$. If there are multiple minimizing solutions, the one with the smallest 2-norm $||x||$ is returned.

NumPy chooses the shortest vector

Parameters: **a** : *(M, N) array_like*

"Coefficient" matrix.

b : *{(M,), (M, K)} array_like*

Ordinate or "dependent variable" values. If *b* is two-dimensional, the least-squares solution is calculated for each of the *K* columns of *b*.

rcond : *float, optional*

Recall: Least Squares in NumPy

numpy.linalg.lstsq

```
linalg.lstsq(a, b, rcond='warn')
```
[source]

Return the least-squares solution to a linear matrix equation.

Computes the vector *x* that approximately solves the equation `a @ x = b`. The equation may be under-, well-, or over-determined (i.e., the number of linearly independent rows of *a* can be less than, equal to, or greater than its number of linearly independent columns). If *a* is square and of full rank, then *x* (but for round-off error) is the "exact" solution of the equation. Else, *x* minimizes the Euclidean 2-norm $||b - ax||$. If there are multiple minimizing solutions, the one with the smallest 2-norm $||x||$ is returned.

NumPy chooses the shortest vector

(why?...)

Parameters: **a** : *(M, N) array_like*

"Coefficient" matrix.

b : *{(M,), (M, K)} array_like*

Ordinate or "dependent variable" values. If *b* is two-dimensional, the least-squares solution is calculated for each of the *K* columns of *b*.

rcond : *float, optional*

Recall: Least Squares in NumPy

numpy.linalg.lstsq

`linalg.lstsq(a, b, rcond='warn')` [source]

Return the least-squares solution to a linear matrix equation.

Computes the vector *x* that approximately solves the equation `a @ x = b`. The equation may be under-, well-, or over-determined (i.e., the number of linearly independent rows of *a* can be less than, equal to, or greater than its number of linearly independent columns). If *a* is square and of full rank, then *x* (but for round-off error) is the "exact" solution of the equation. Else, *x* minimizes the Euclidean 2-norm $||b - ax||$. If there are multiple minimizing solutions, the one with the smallest 2-norm $||x||$ is returned.

NumPy chooses the shortest vector

(why?...)

because they use SVD!

Parameters: **a** : *(M, N) array_like*

"Coefficient" matrix.

b : *{(M,), (M, K)} array_like*

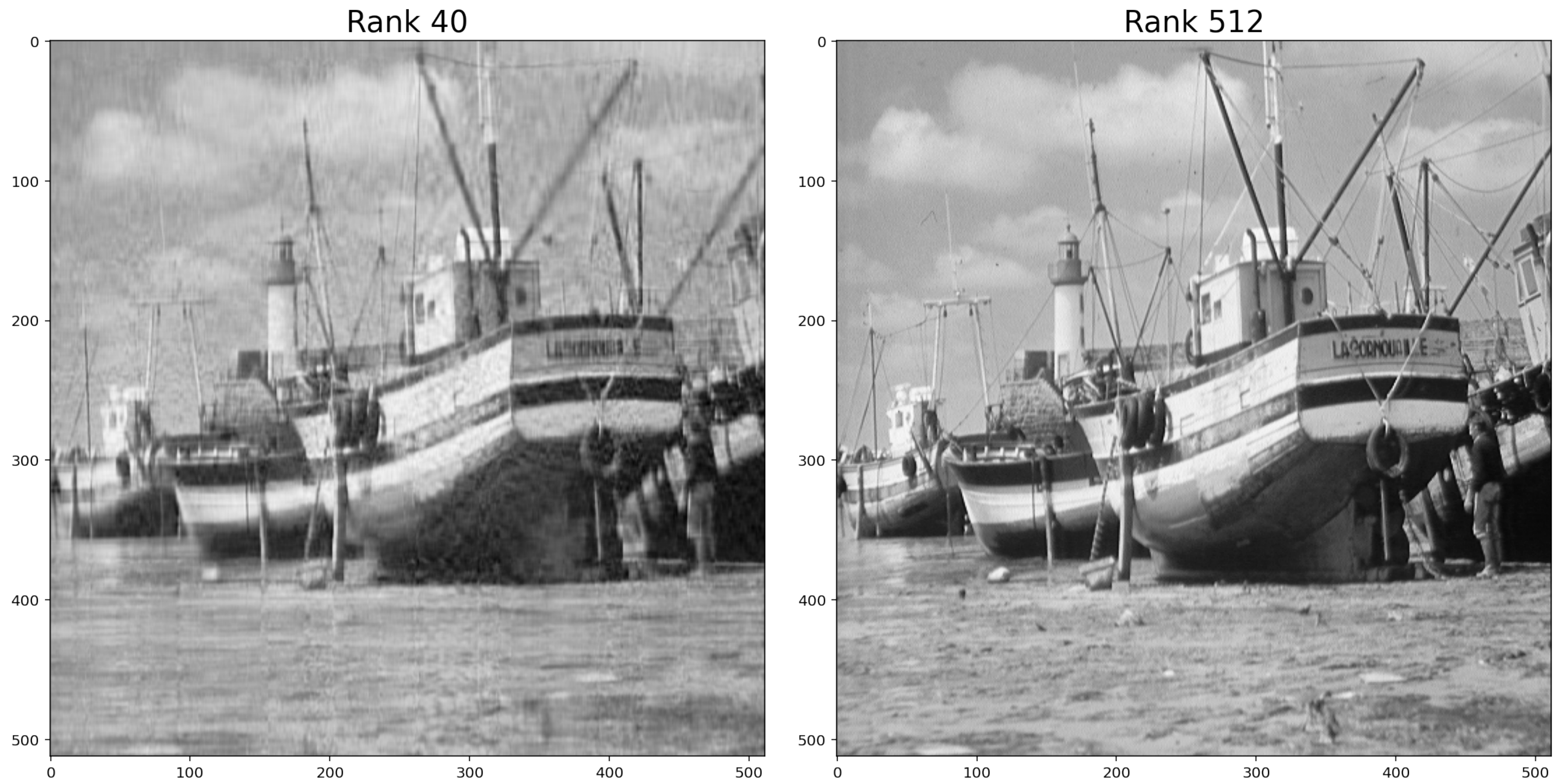
Ordinate or "dependent variable" values. If *b* is two-dimensional, the least-squares solution is calculated for each of the *K* columns of *b*.

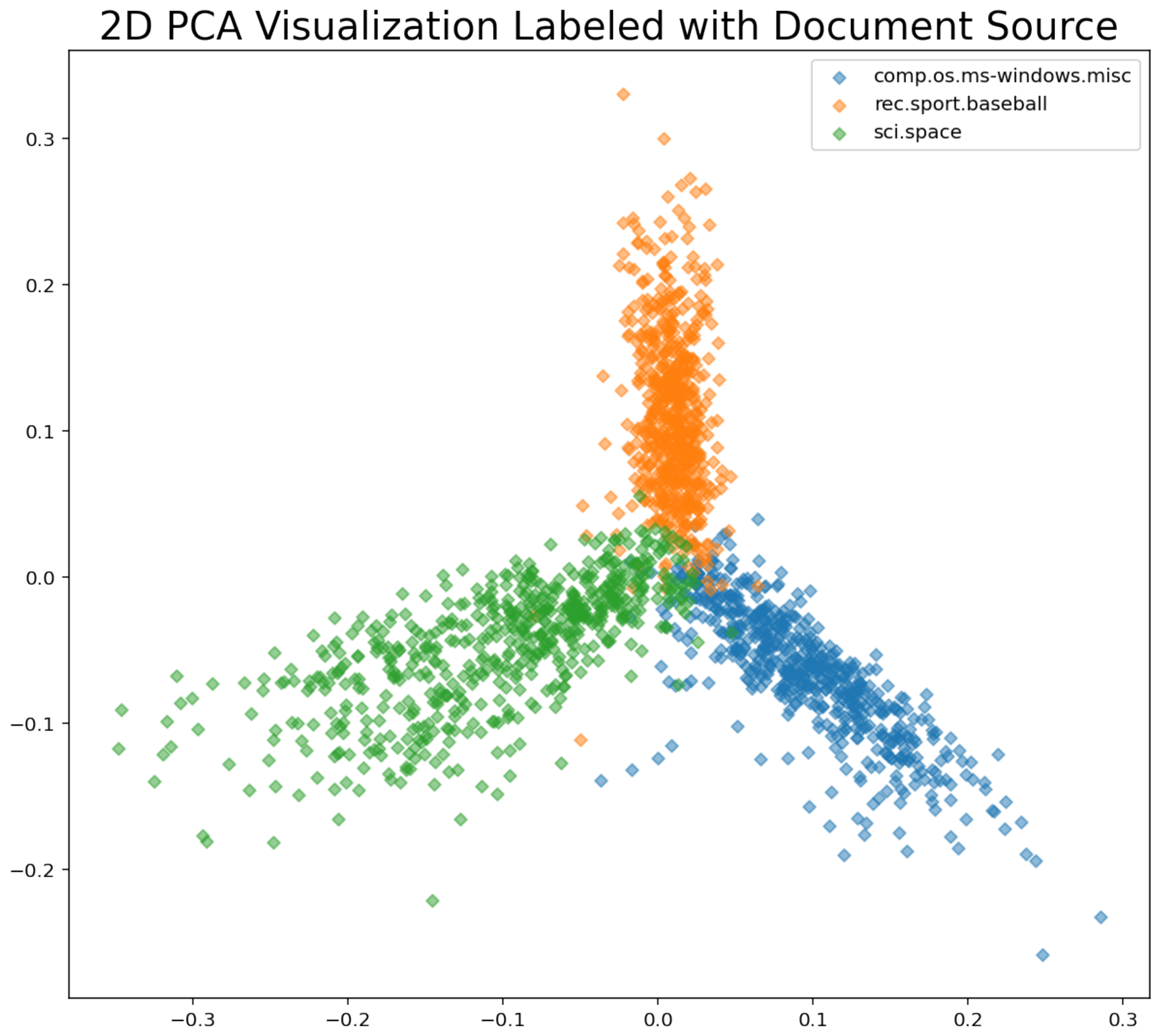
rcond : *float, optional*

What's next?
A couple final thoughts

Applications of SVD

image compression



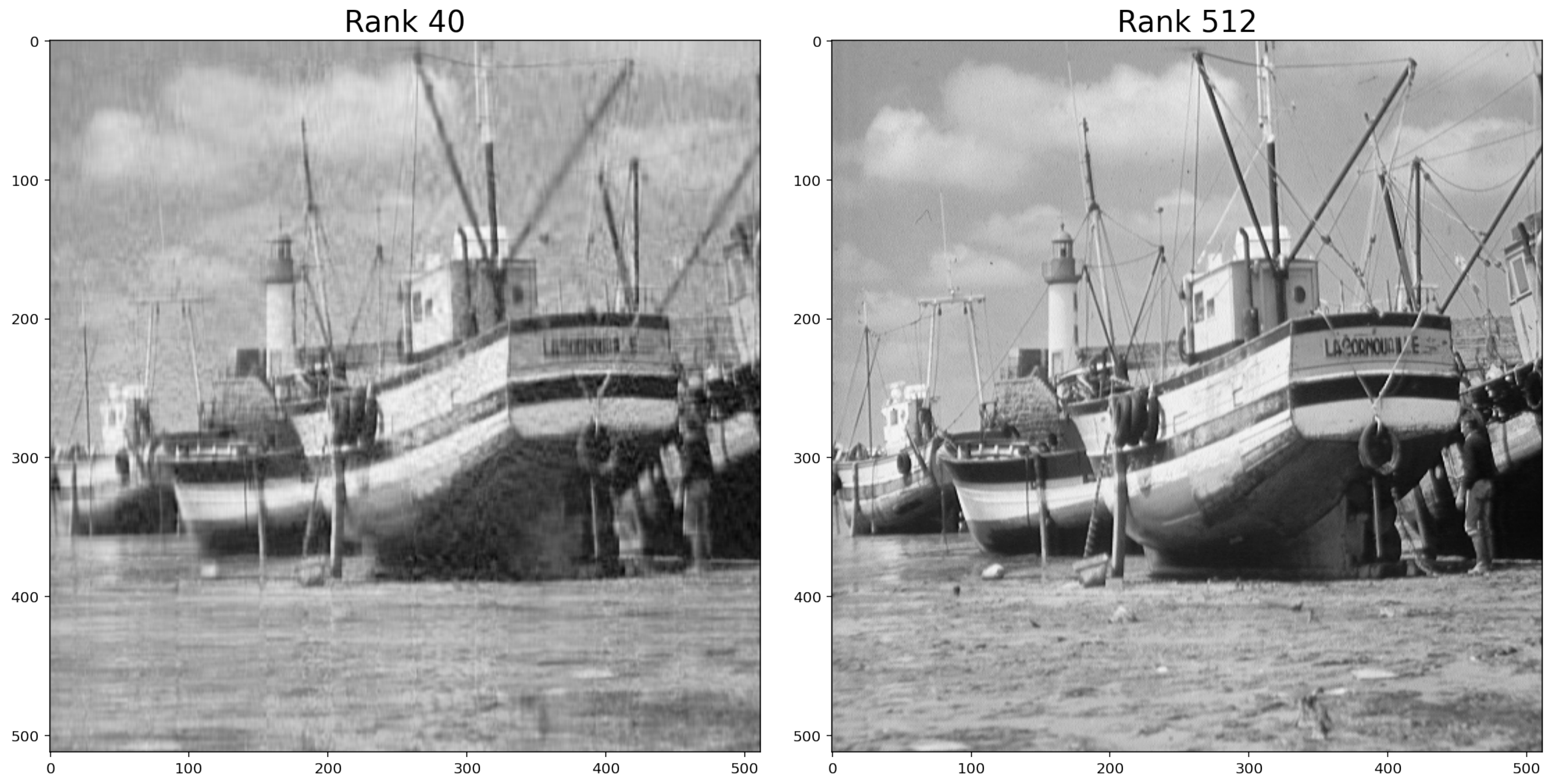


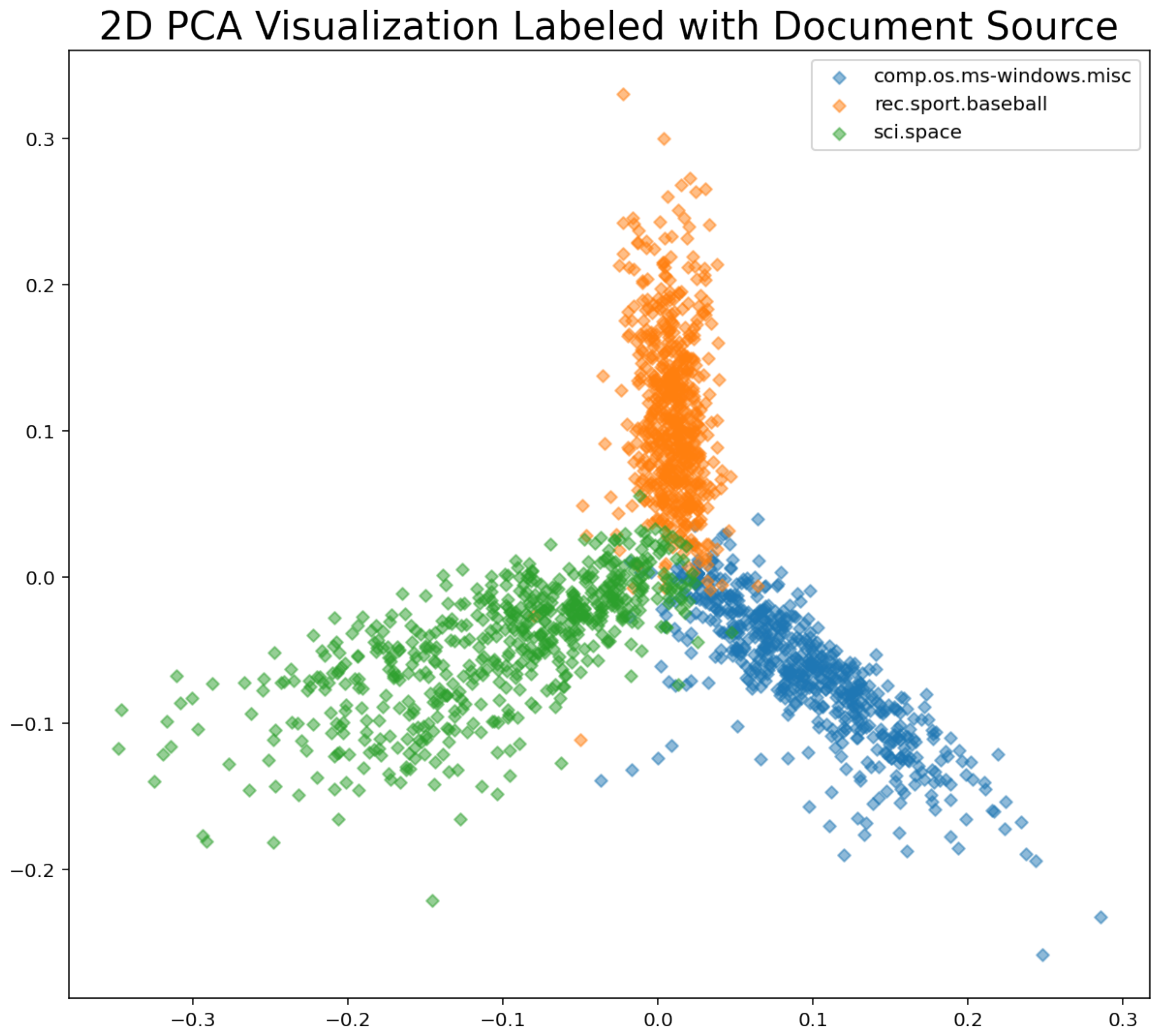
document classification

Applications of SVD

- Reduced SVD, pseudoinverses and least squares

image compression



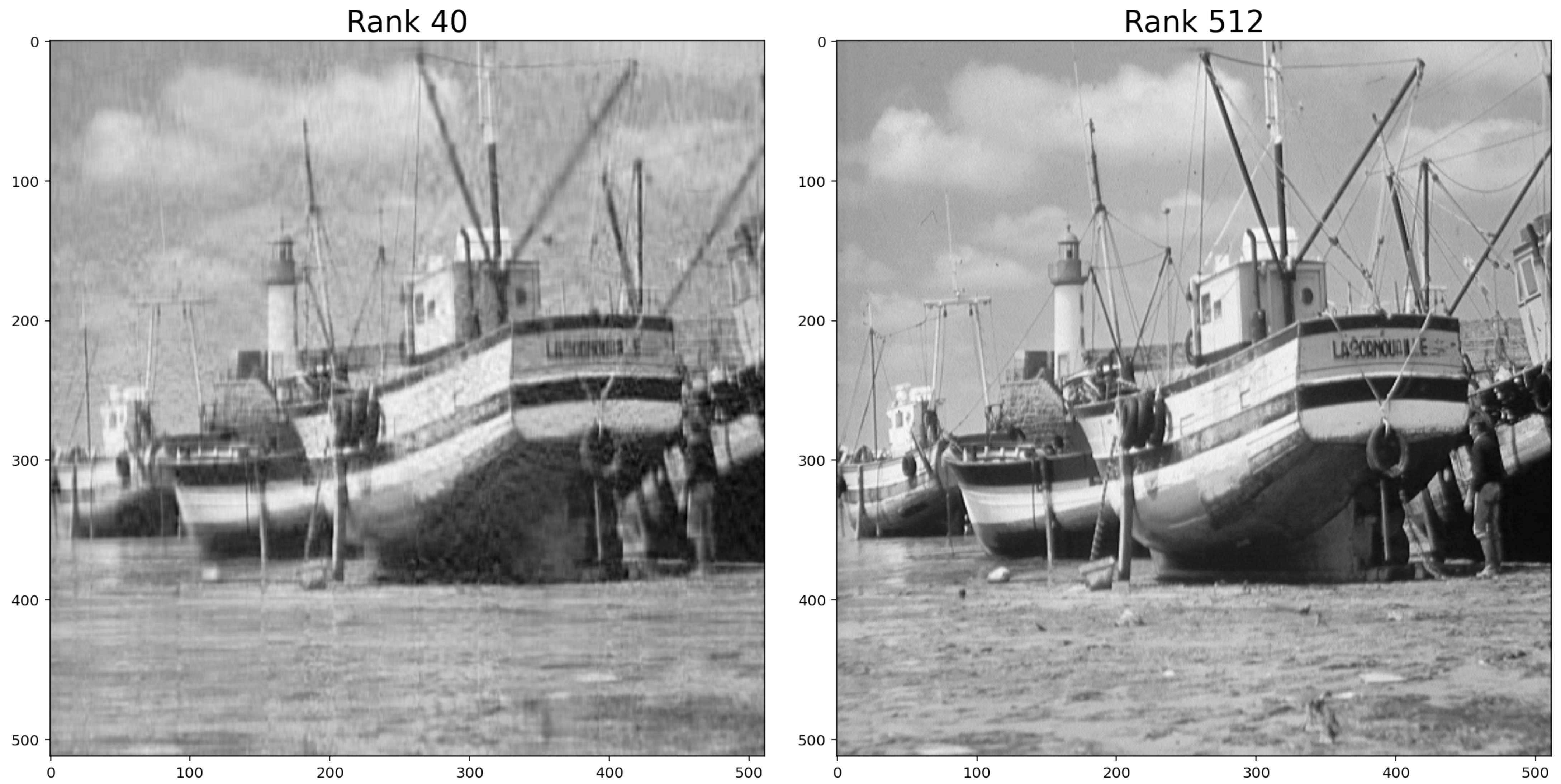


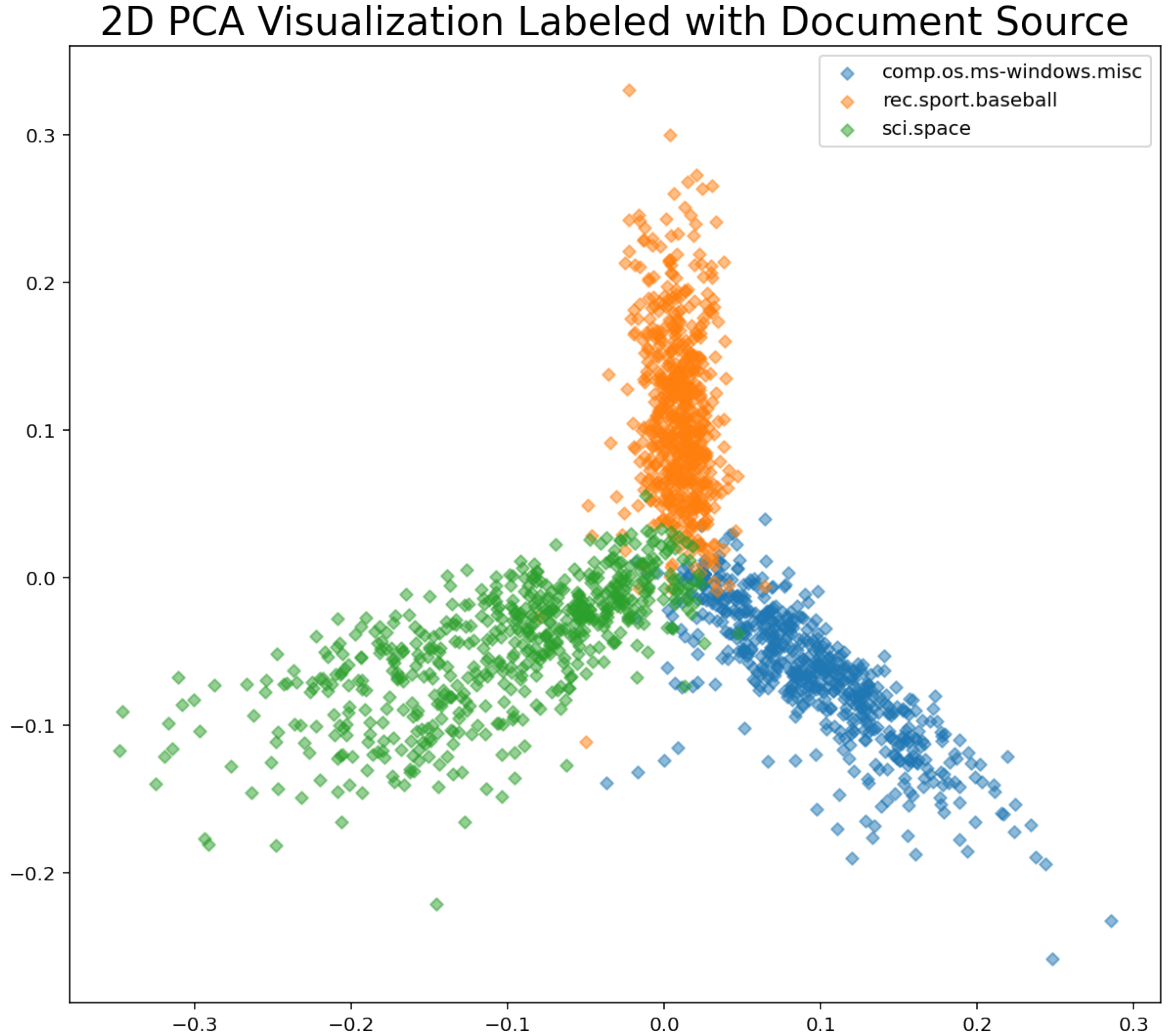
document classification

Applications of SVD

- Reduced SVD, pseudoinverses and least squares
  - If $A^+ = V\Sigma^{-1}U^T$, then $A^+\mathbf{b}$ is a least squares solution of minimum length

image compression



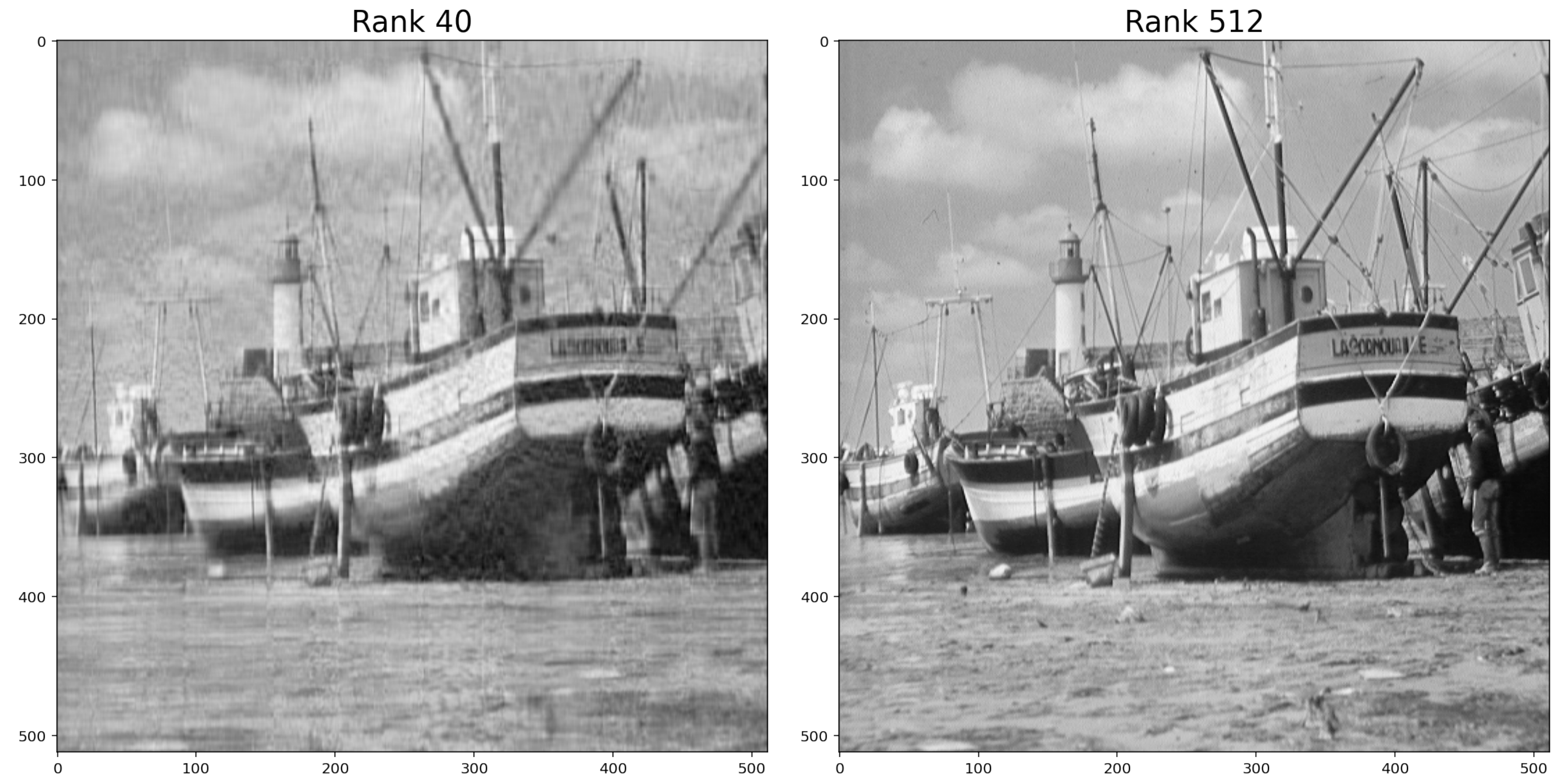


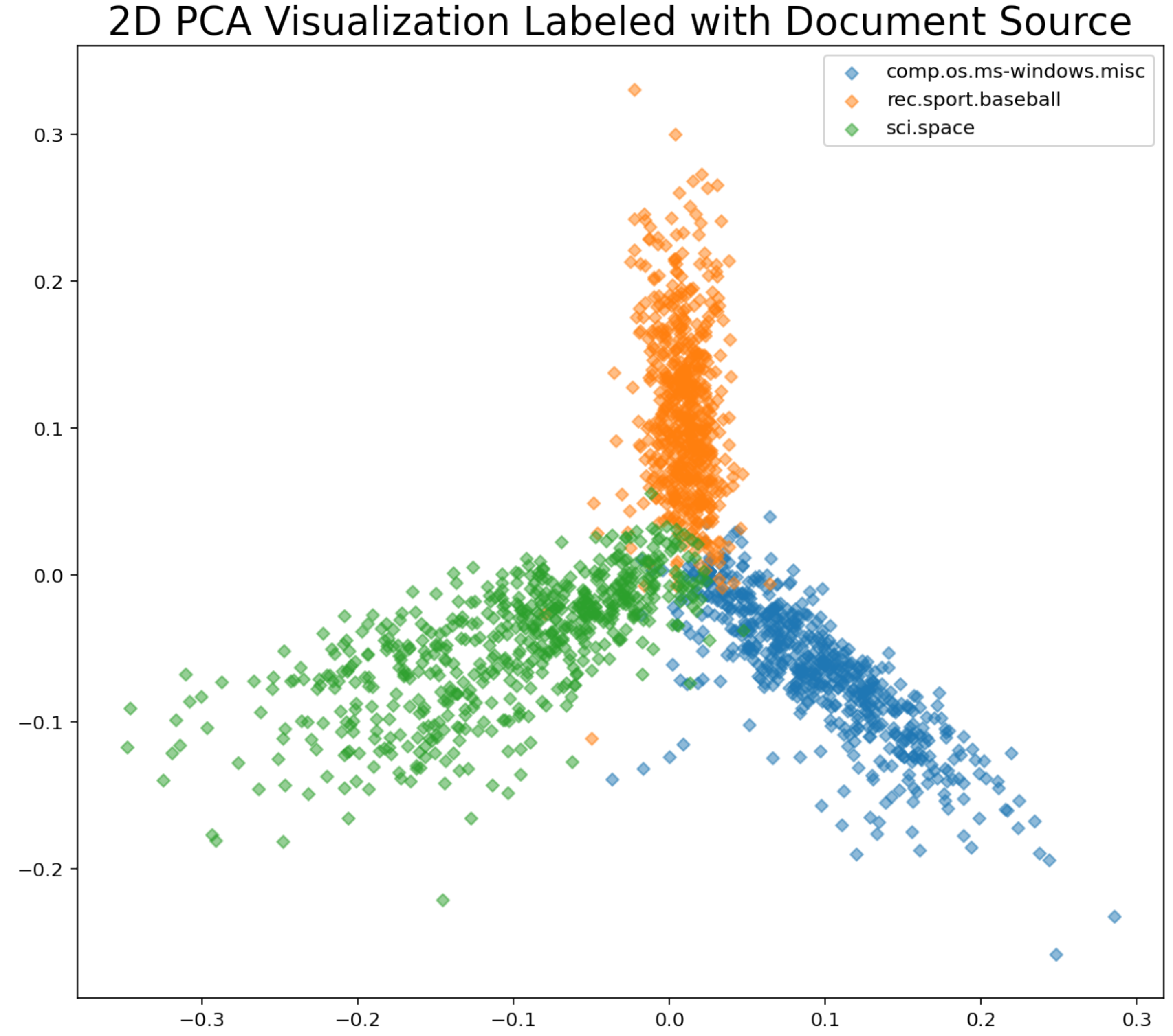
document classification

Applications of SVD

- Reduced SVD, pseudoinverses and least squares
  - If $A^+ = V\Sigma^{-1}U^T$, then $A^+\mathbf{b}$ is a least squares solution of minimum length
- Low Rank Approximation and Data Compression

image compression



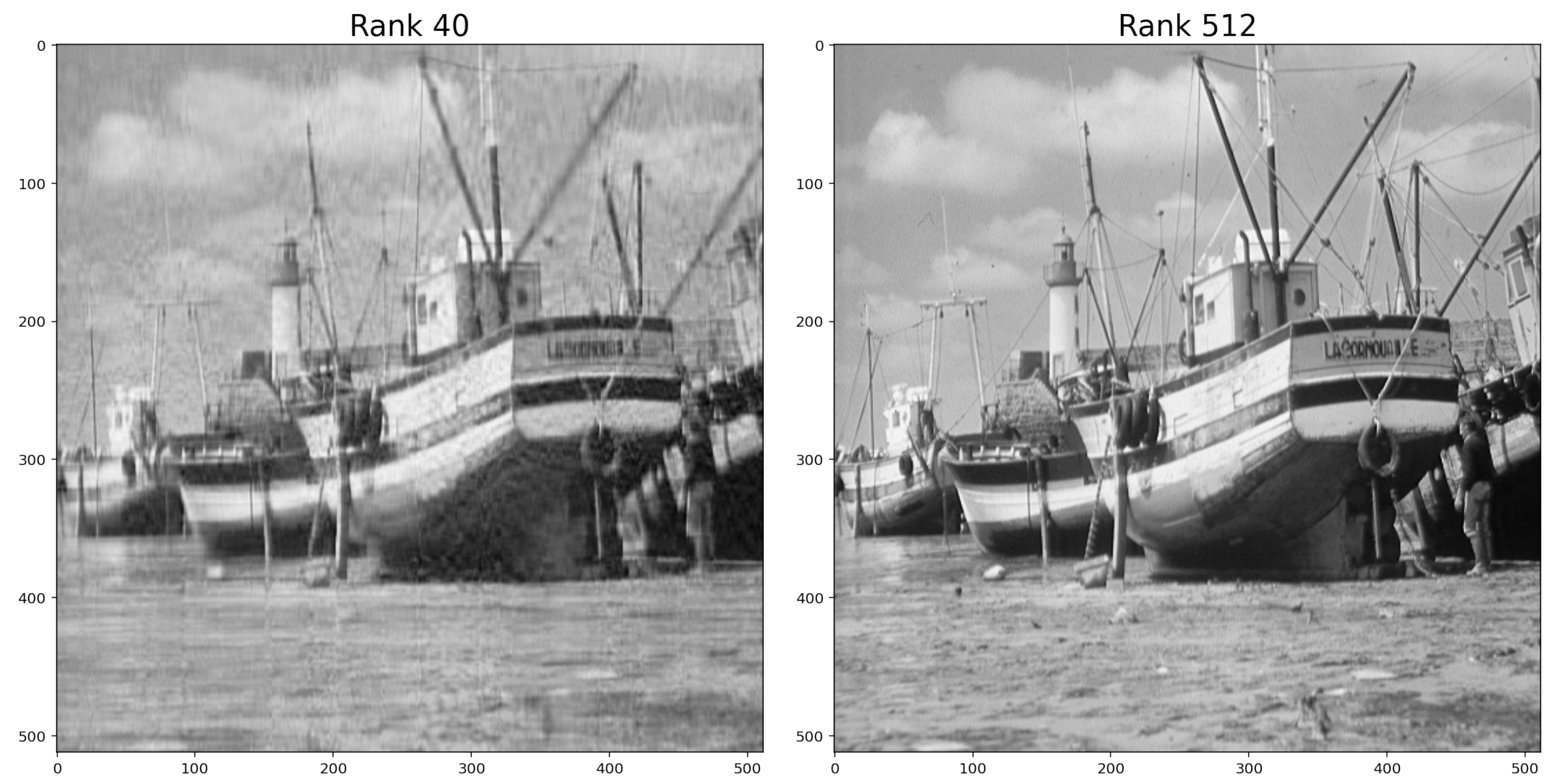


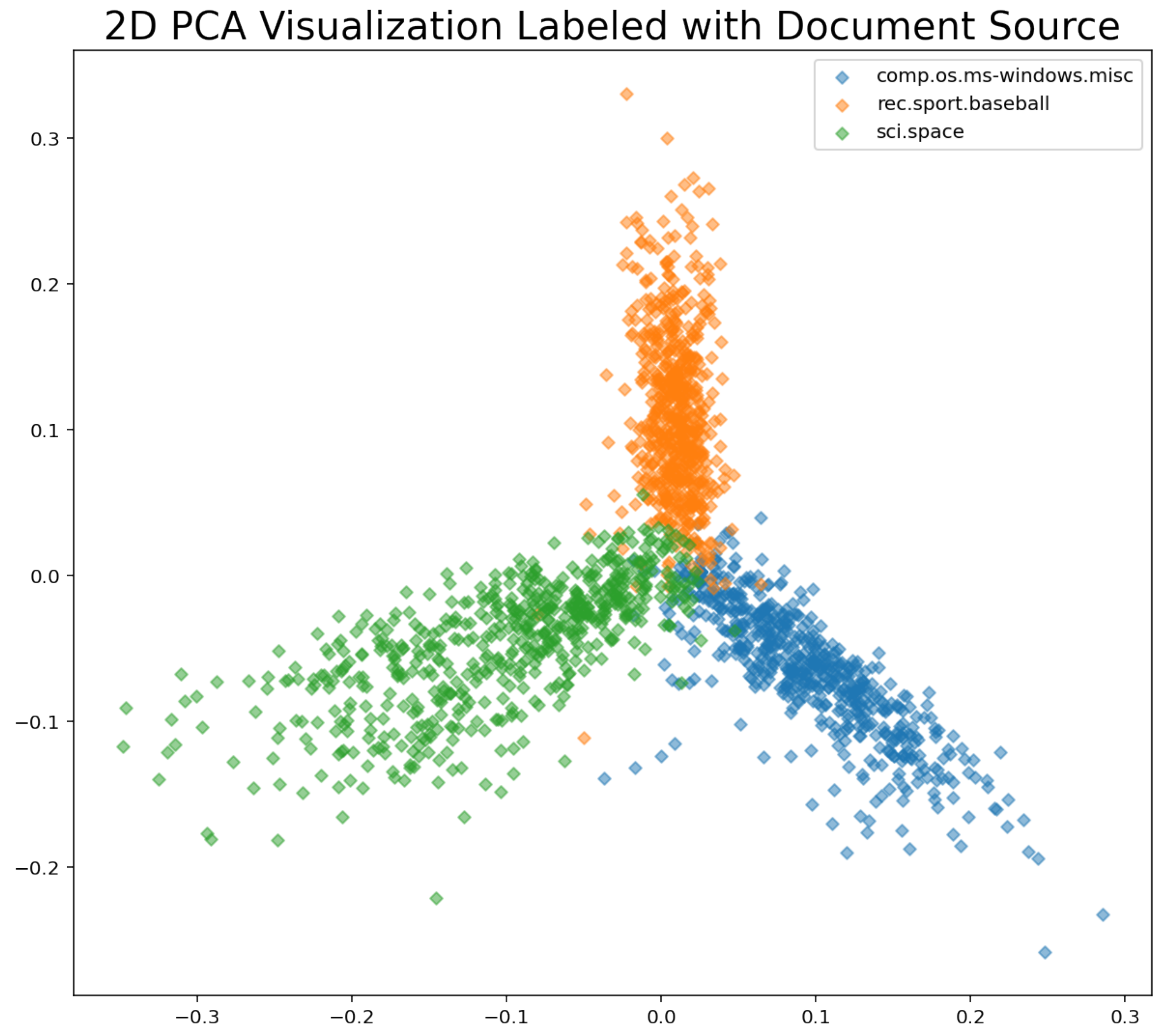
document classification

Applications of SVD

- Reduced SVD, pseudoinverses and least squares
  - If $A^+ = V\Sigma^{-1}U^T$, then $A^+\mathbf{b}$ is a least squares solution of minimum length
- Low Rank Approximation and Data Compression
  - Replacing small singular values with zero in Σ gives a good approximation to A.

image compression



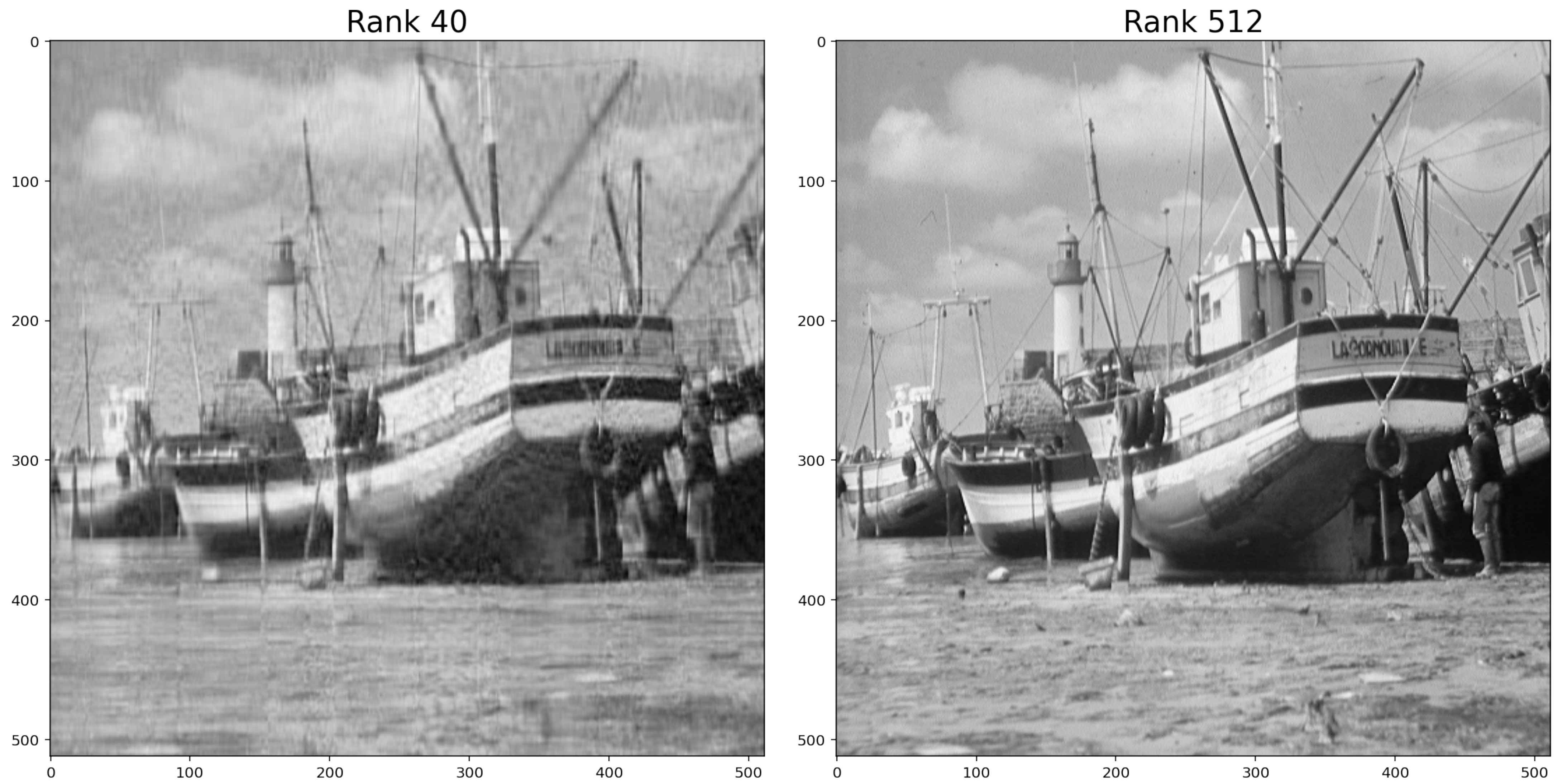


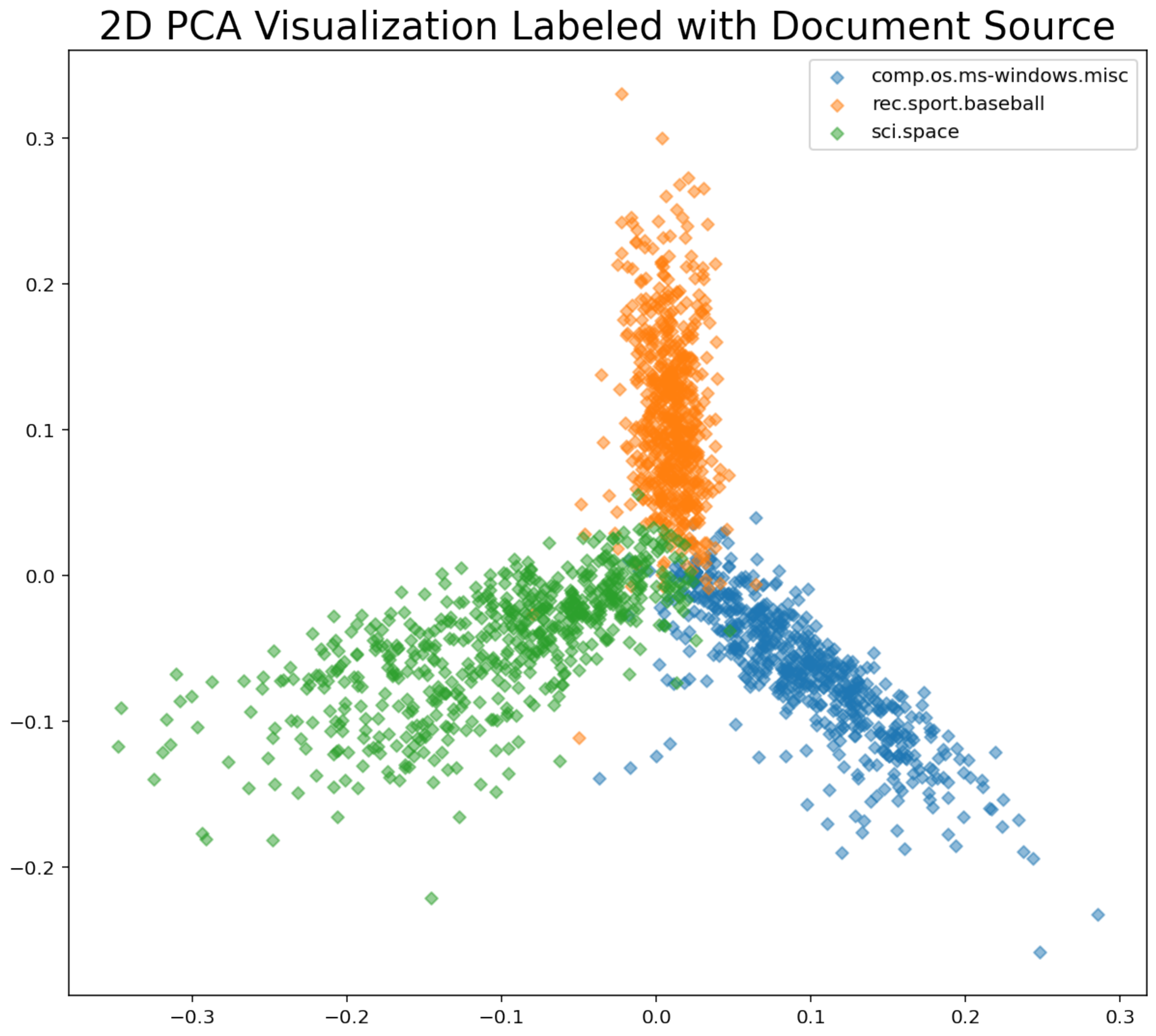
document classification

Applications of SVD

- Reduced SVD, pseudoinverses and least squares
  - If $A^+ = V\Sigma^{-1}U^T$, then $A^+\mathbf{b}$ is a least squares solution of minimum length
- Low Rank Approximation and Data Compression
  - Replacing small singular values with zero in Σ gives a good approximation to A.
  - This is used for image compression

image compression



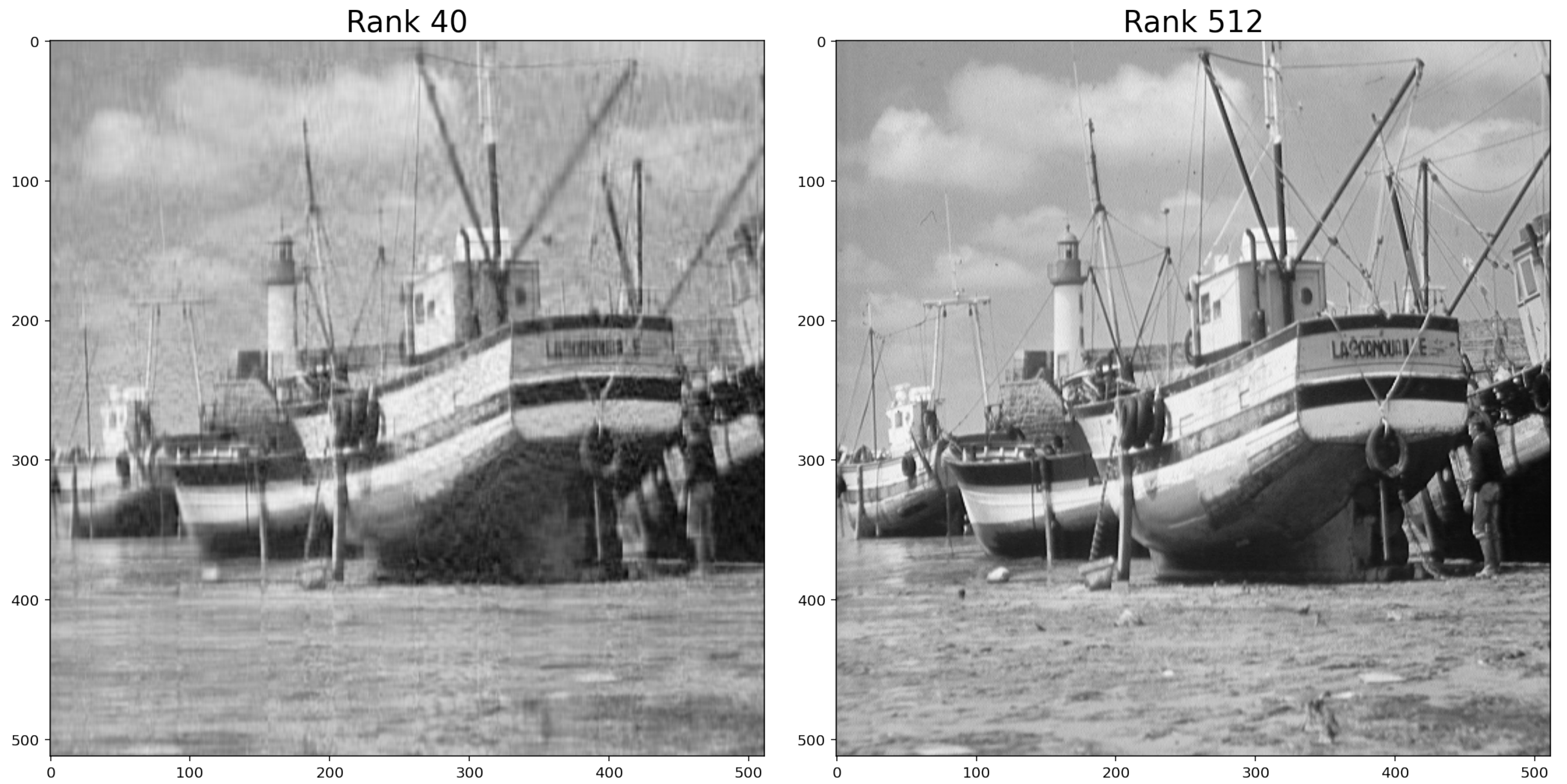


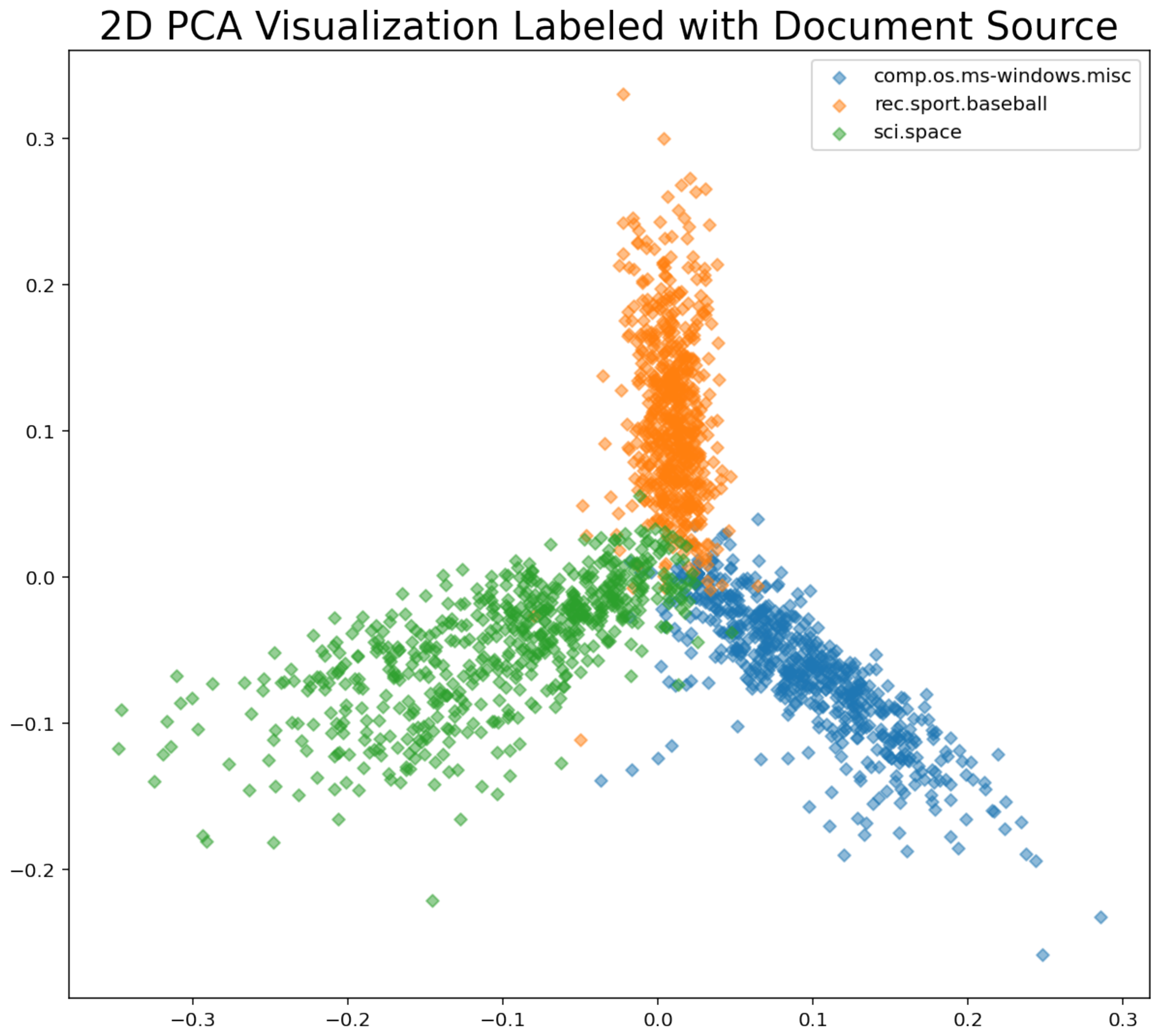
document classification

Applications of SVD

- Reduced SVD, pseudoinverses and least squares
  - If $A^+ = V\Sigma^{-1}U^T$, then $A^+\mathbf{b}$ is a least squares solution of minimum length
- Low Rank Approximation and Data Compression
  - Replacing small singular values with zero in Σ gives a good approximation to A.
  - This is used for image compression
- Principle Component Analysis

image compression



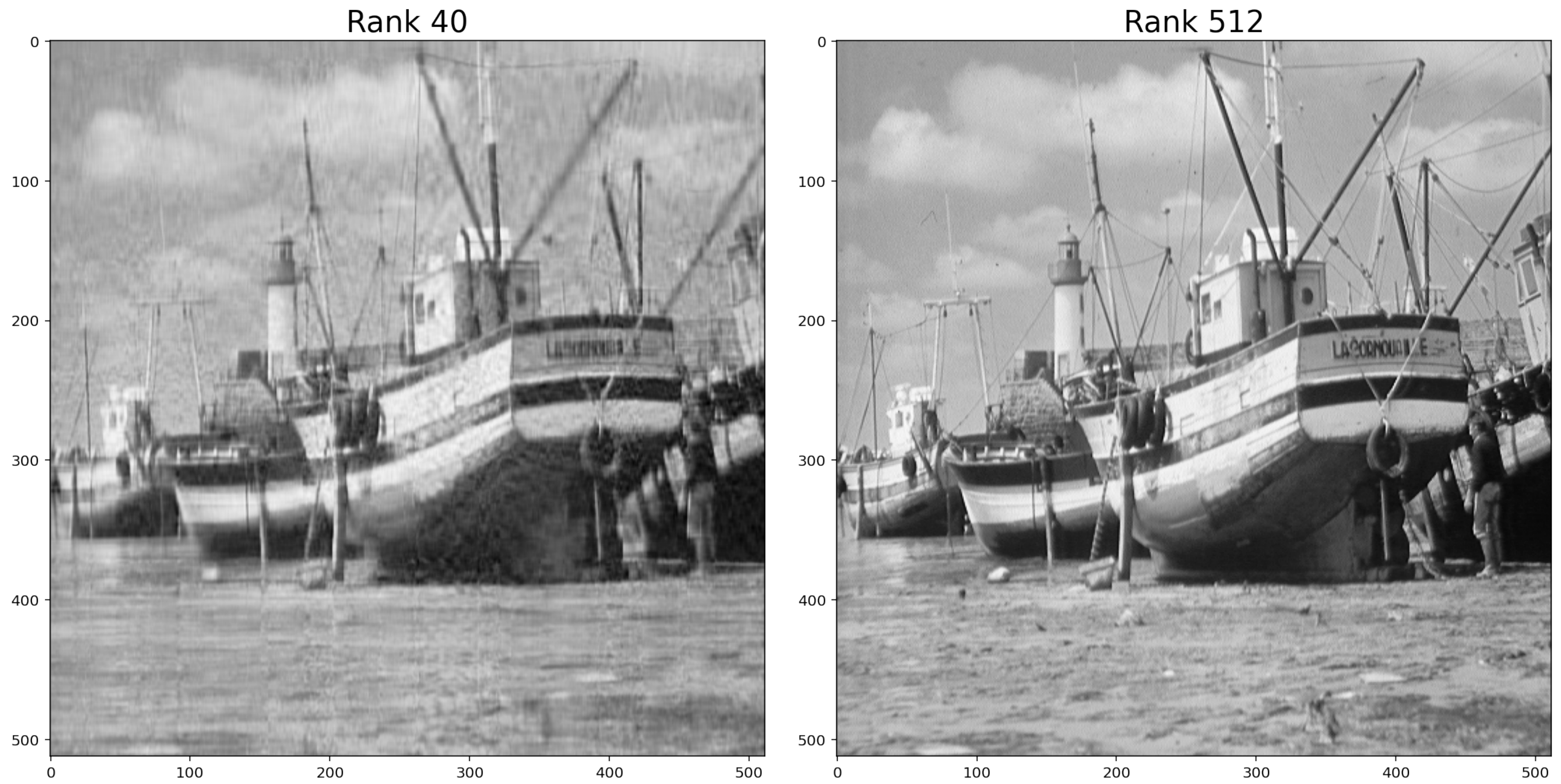


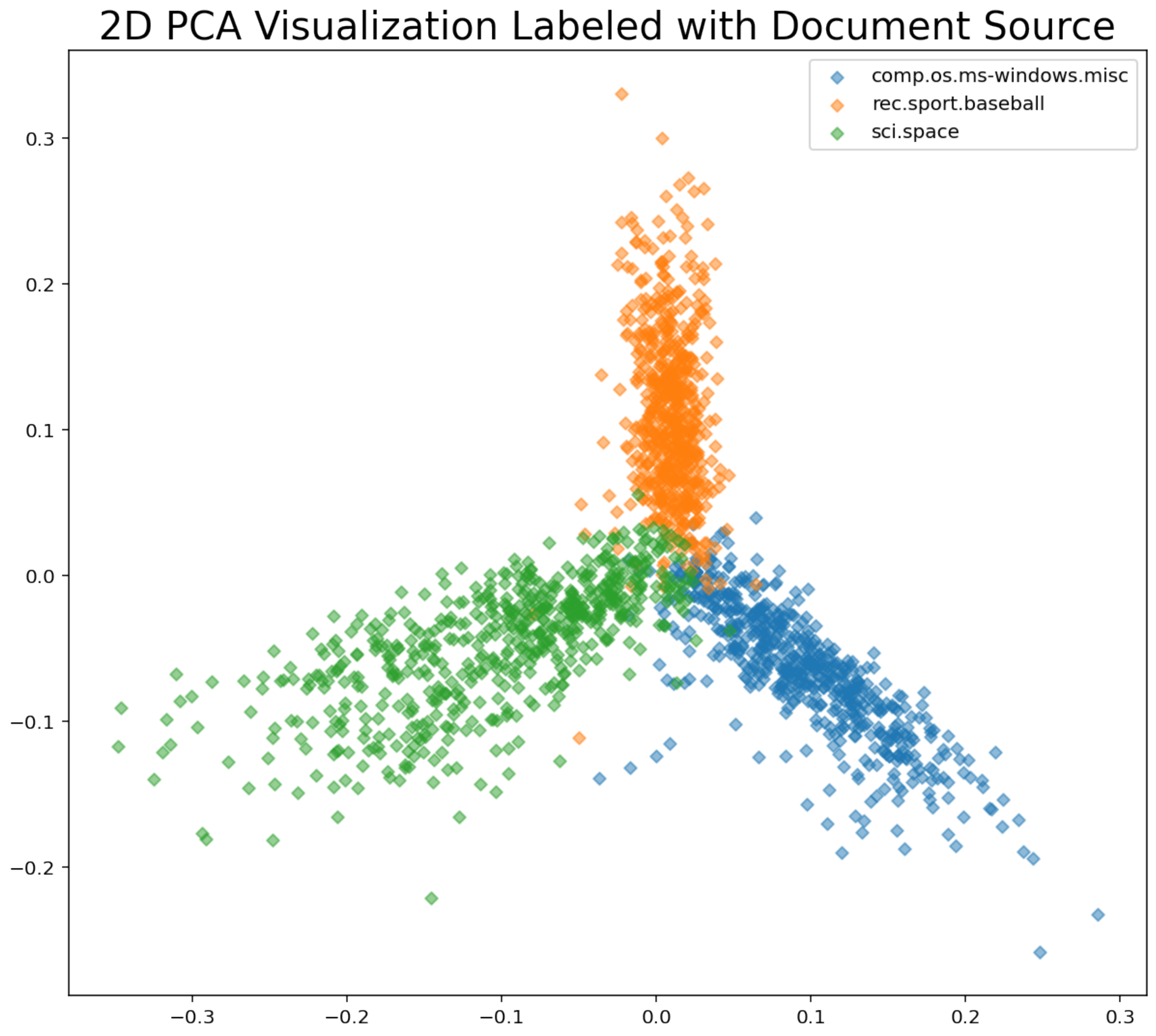
document classification

Applications of SVD

- Reduced SVD, pseudoinverses and least squares
    - If $A^+ = V\Sigma^{-1}U^T$, then $A^+\mathbf{b}$ is a least squares solution of minimum length
- Low Rank Approximation and Data Compression
    - Replacing small singular values with zero in Σ gives a good approximation to A.
    - This is used for image compression
- Principle Component Analysis
    - Large singular vectors are "most affected."

image compression



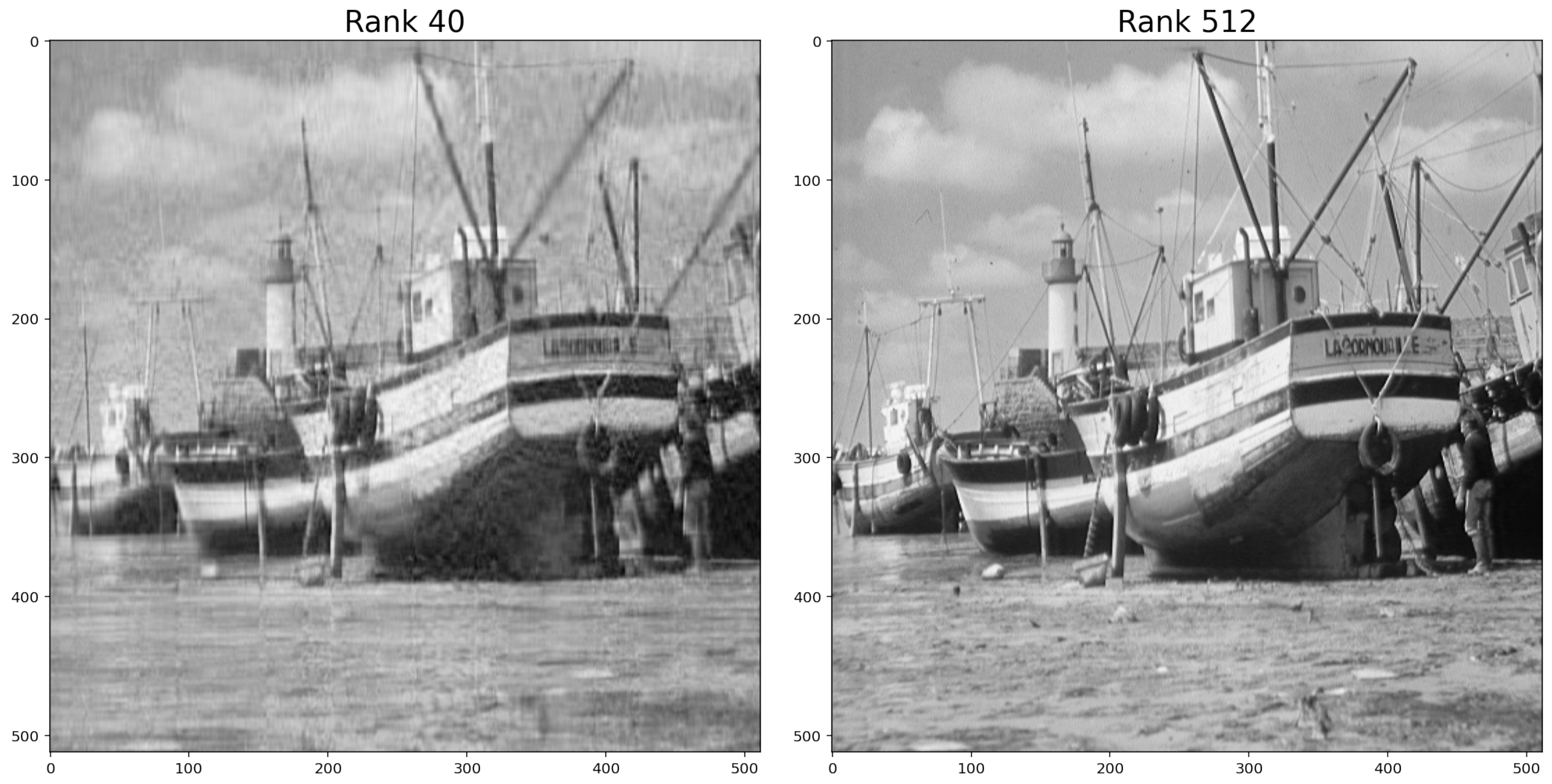


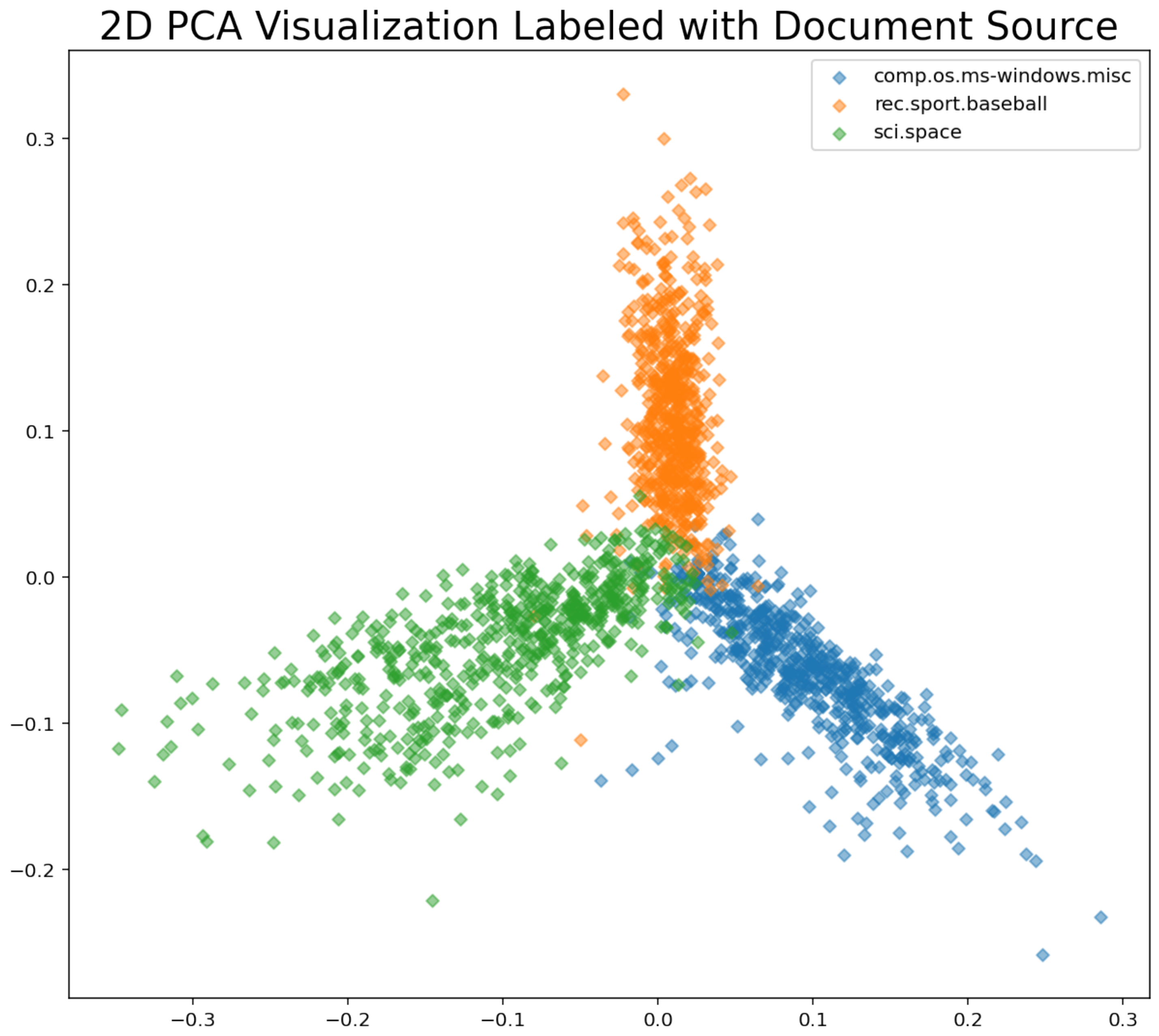
document classification

Applications of SVD

- Reduced SVD, pseudoinverses and least squares
  - If $A^+ = V\Sigma^{-1}U^T$, then $A^+\mathbf{b}$ is a least squares solution of minimum length
- Low Rank Approximation and Data Compression
  - Replacing small singular values with zero in Σ gives a good approximation to A.
  - This is used for image compression
- Principle Component Analysis
  - Large singular vectors are "most affected."
  - These are good vectors to look at for classifying data

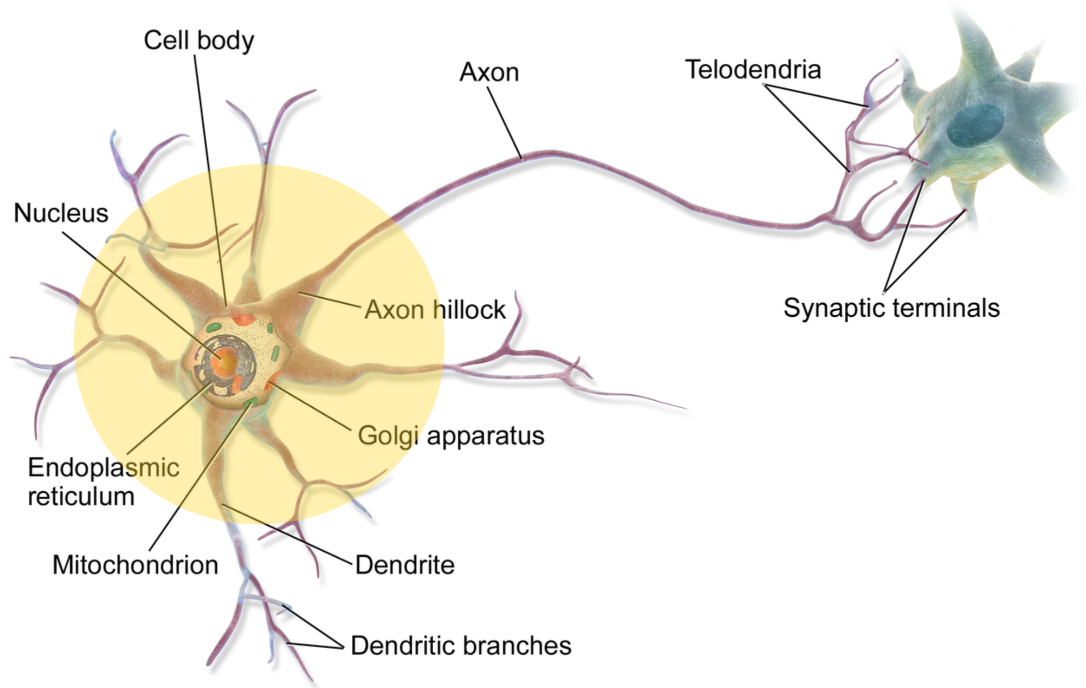
image compression

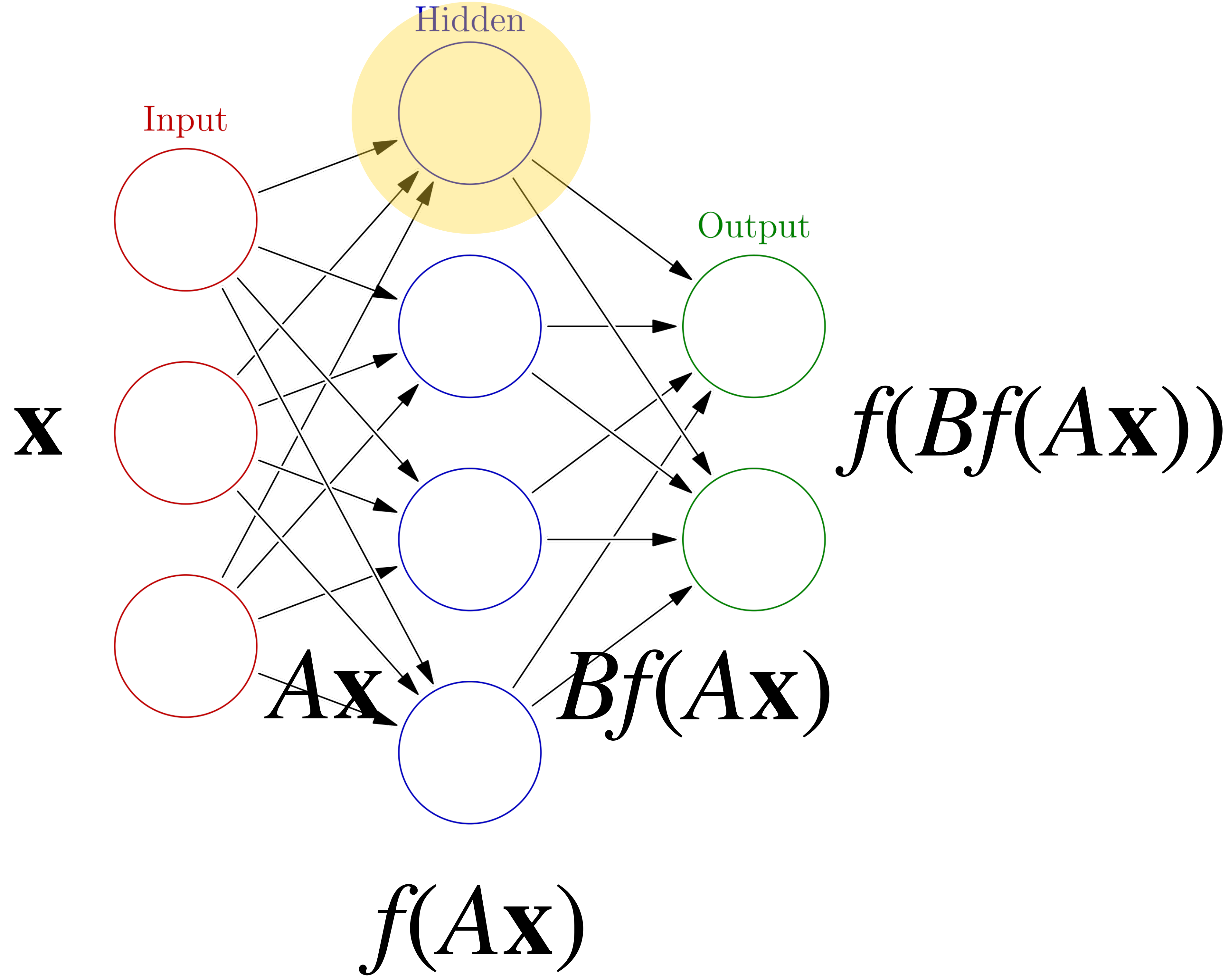




document classification

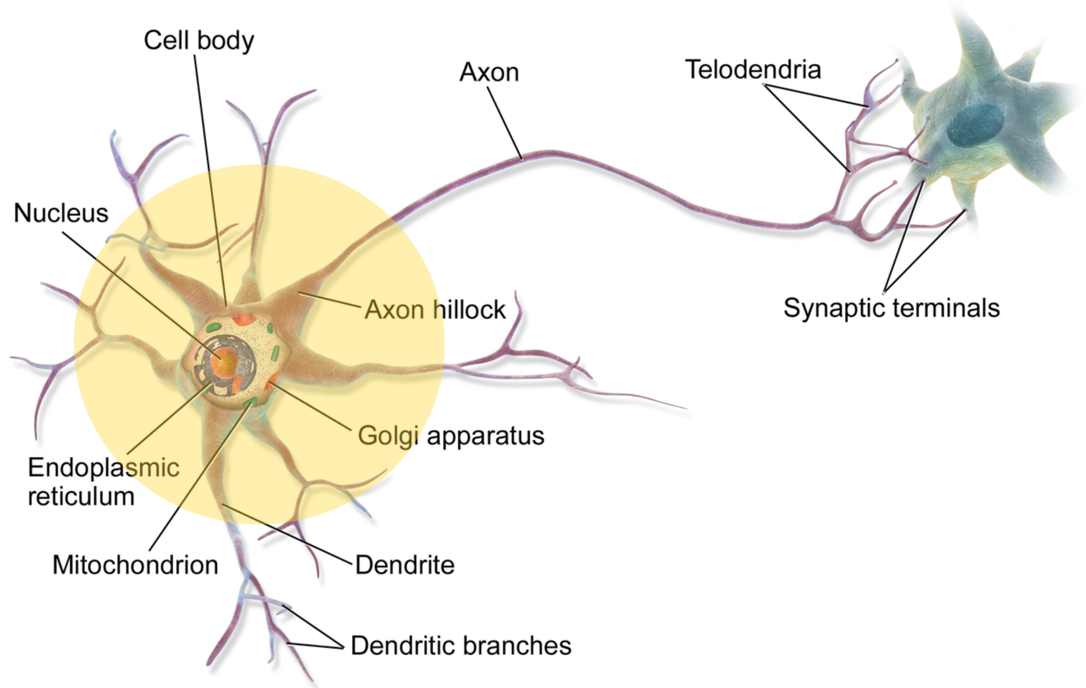
Neural Networks (Non-Linearity)



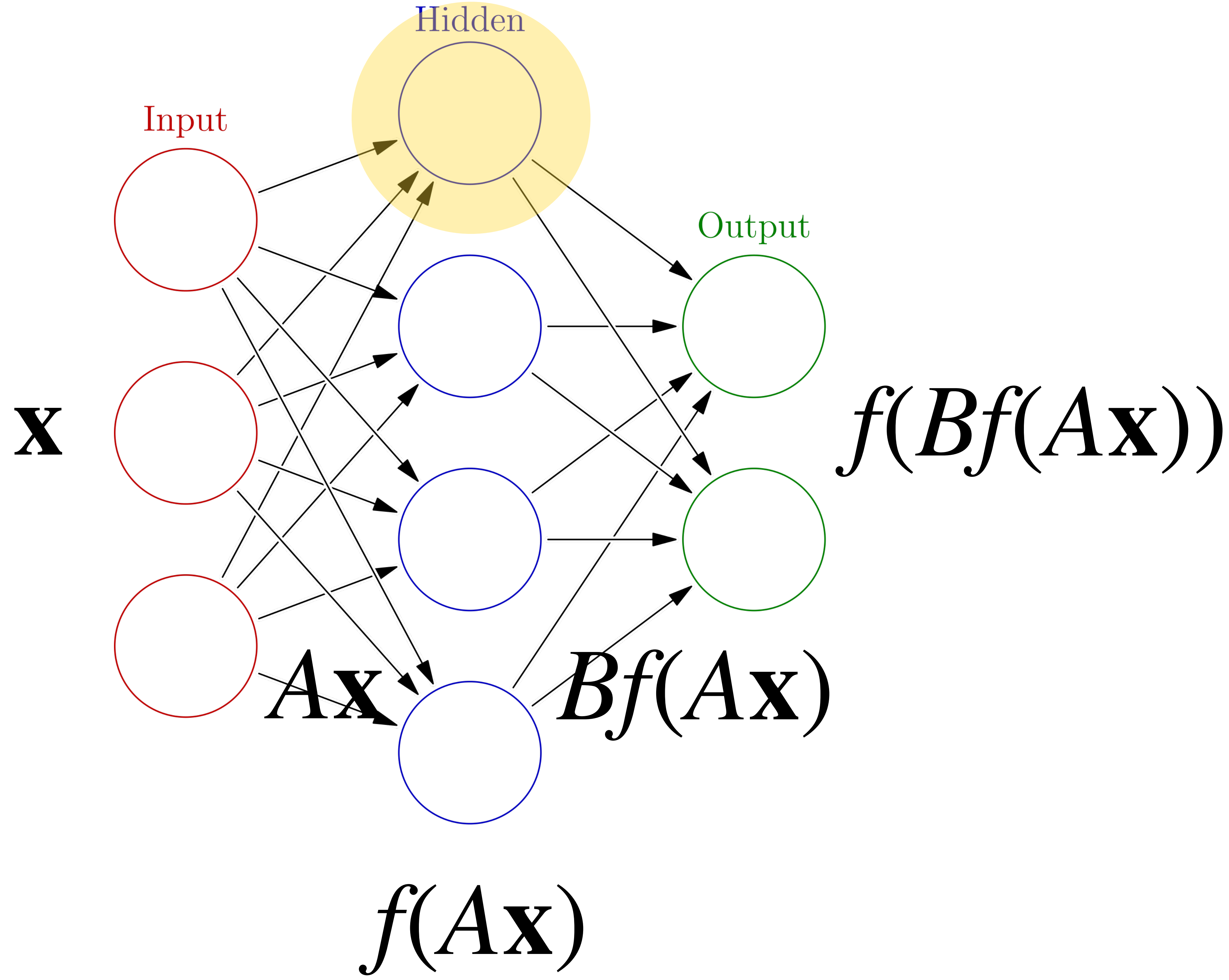


https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png
https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

Neural Networks (Non-Linearity)

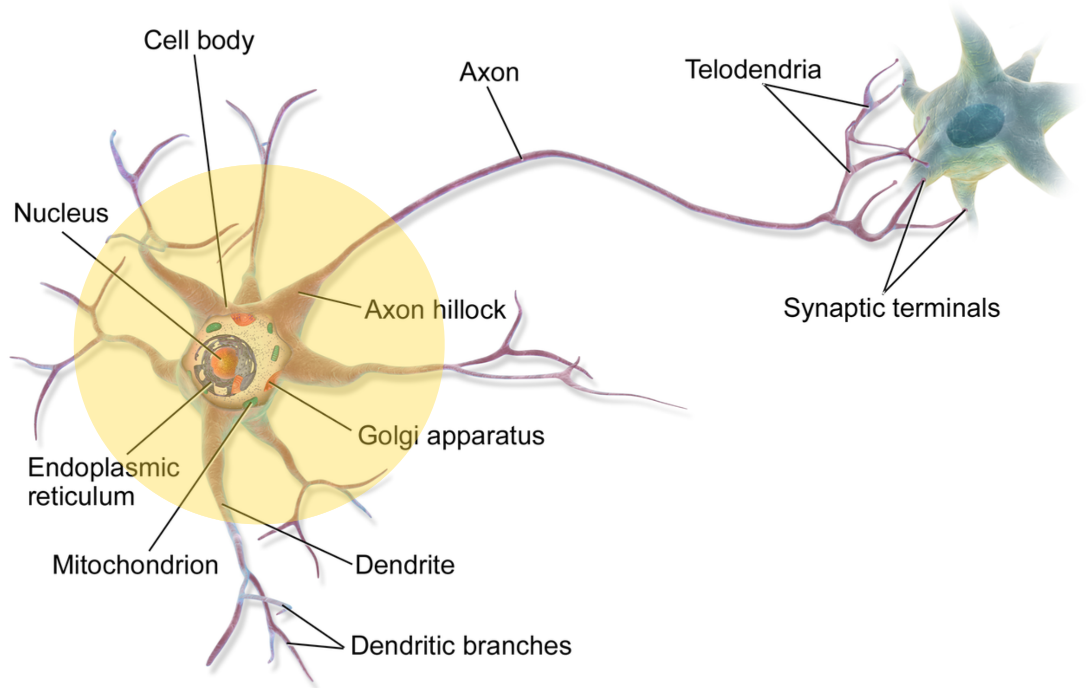


Neural networks are models of artificial neurons bundles.



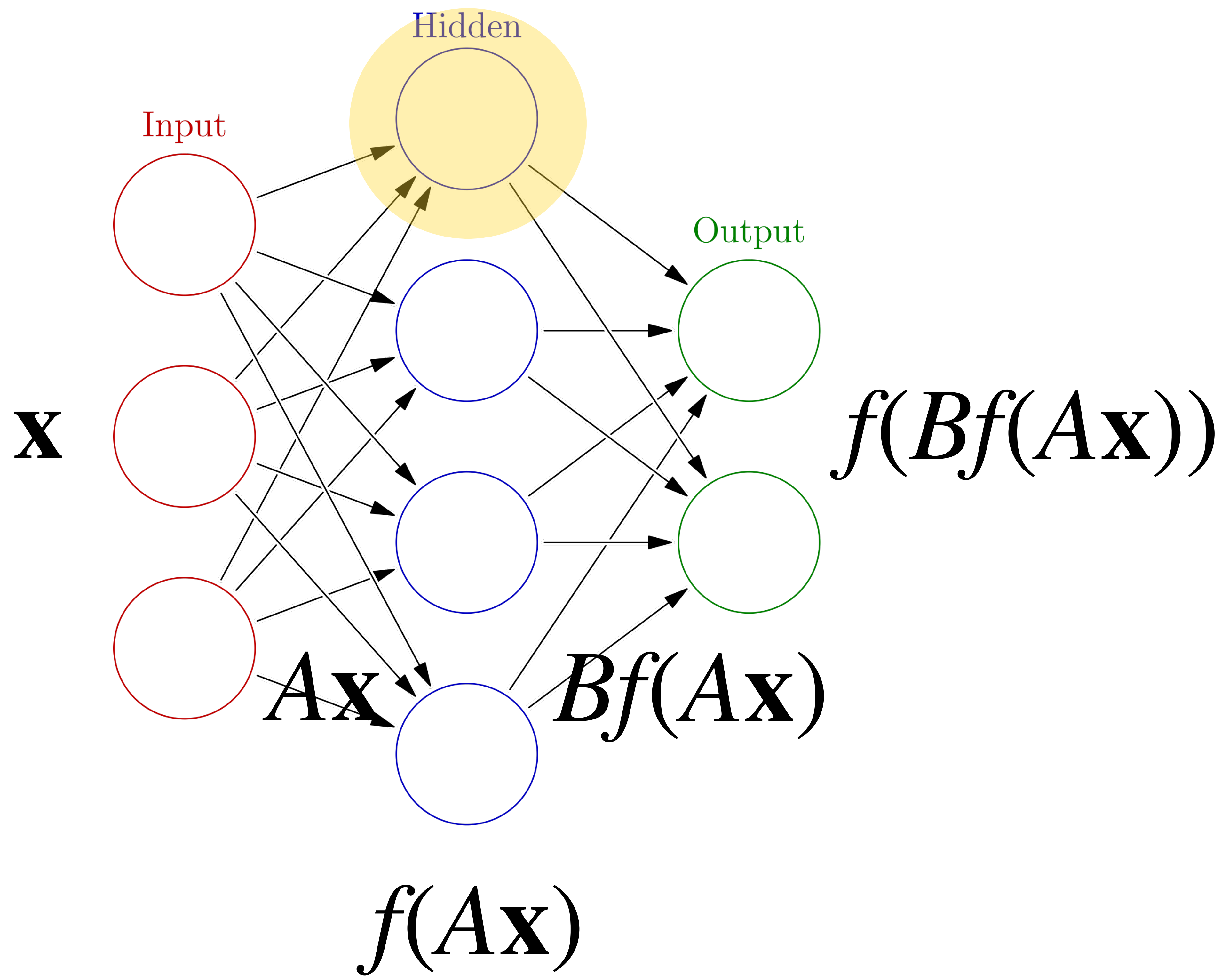
https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png
https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

Neural Networks (Non-Linearity)



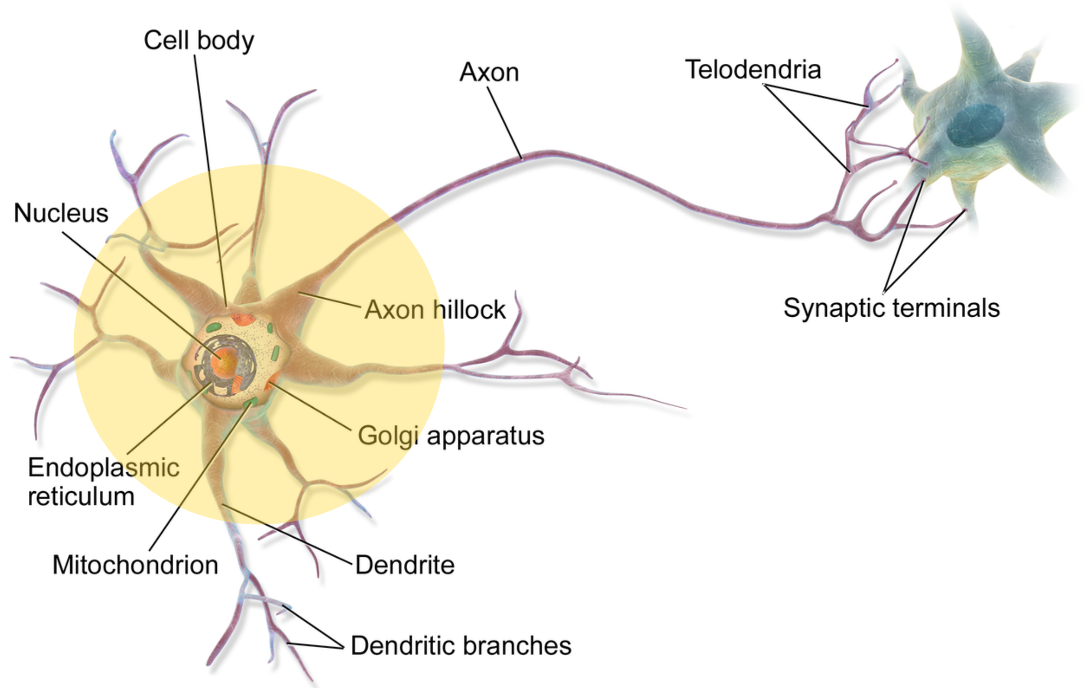
Neural networks are models of artificial neurons bundles.

Given an input vector $\mathbf{x}$, it is transformed into a *hidden* vector $A\mathbf{x}$ by a linear transformation, and then an *activation function* f is applied to the result.



https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png
https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

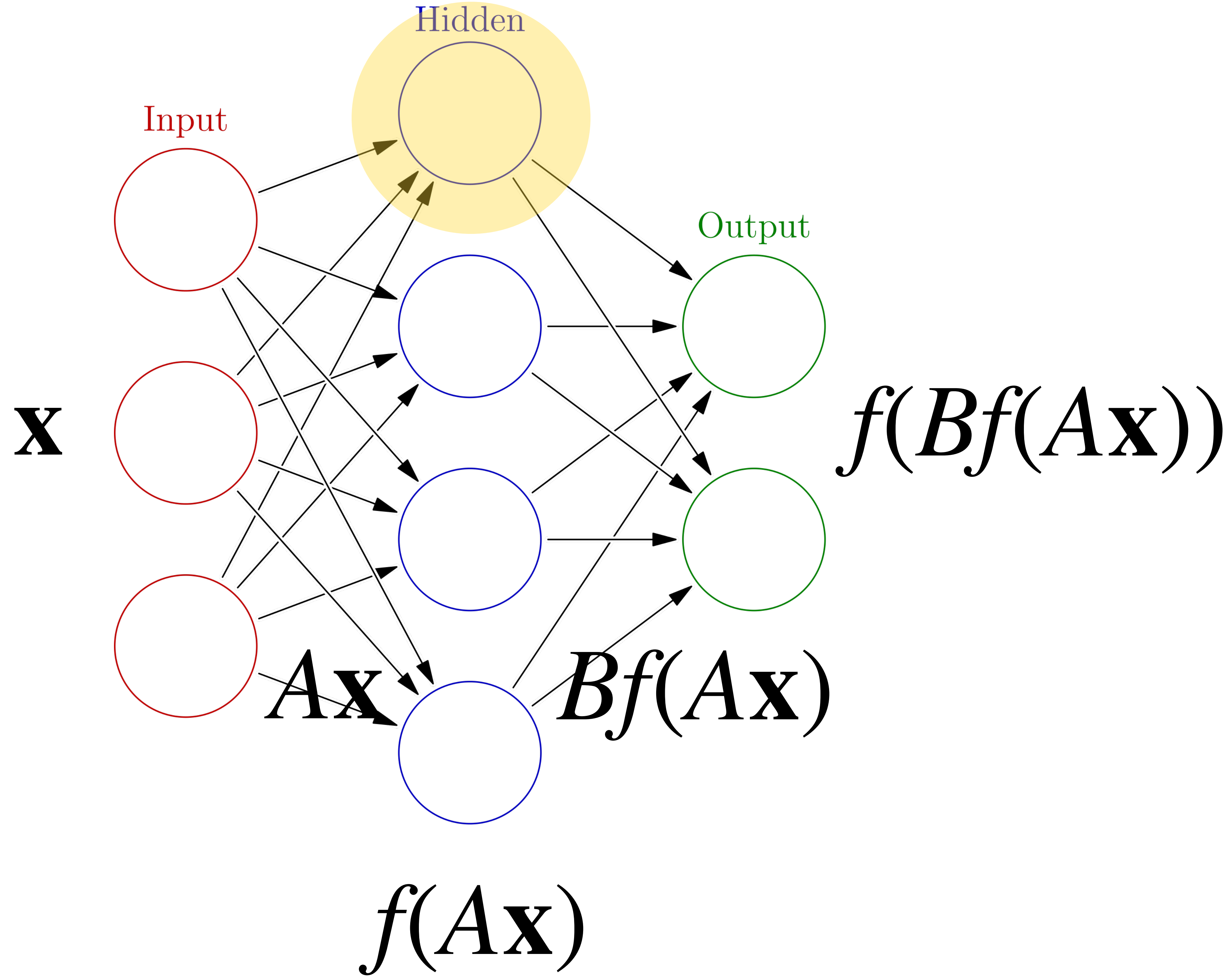
Neural Networks (Non-Linearity)



Neural networks are models of artificial neurons bundles.

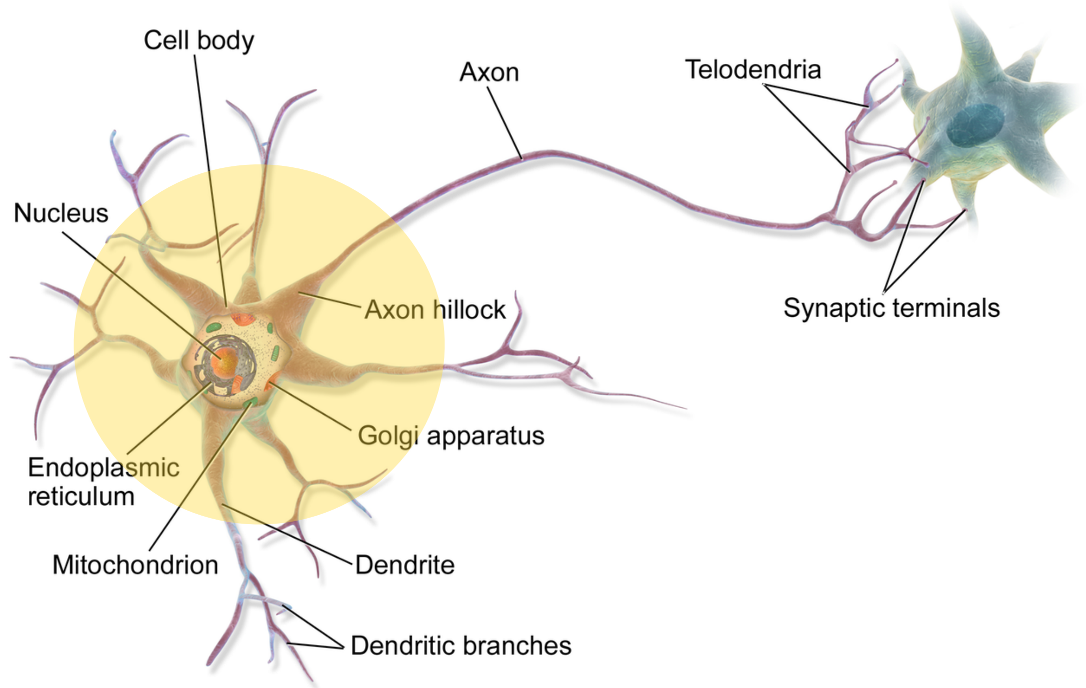
Given an input vector $\mathbf{x}$, it is transformed into a *hidden* vector $A\mathbf{x}$ by a linear transformation, and then an *activation function* f is applied to the result.

Neural networks are just matrix multiplications with intermediate calls to a nonlinear function f.



https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png
https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

Neural Networks (Non-Linearity)

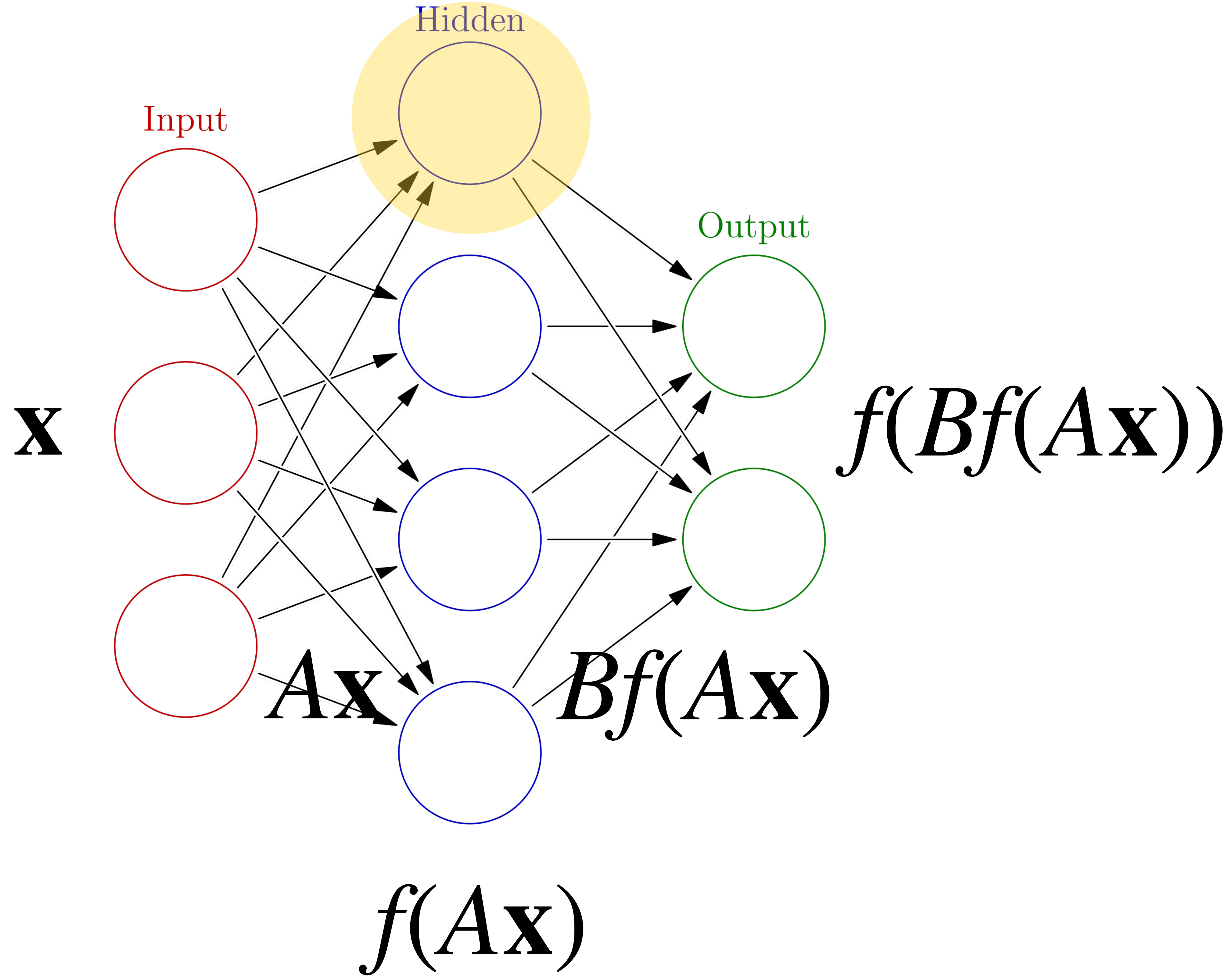


Neural networks are models of artificial neurons bundles.

Given an input vector $\mathbf{x}$, it is transformed into a *hidden* vector $A\mathbf{x}$ by a linear transformation, and then an *activation function* f is applied to the result.

Neural networks are just matrix multiplications with intermediate calls to a nonlinear function f.

$$\mathrm{NN}(\mathbf{x}) = f(A_k(f(A_{k-1} \ldots f(A_1\mathbf{x}))$$

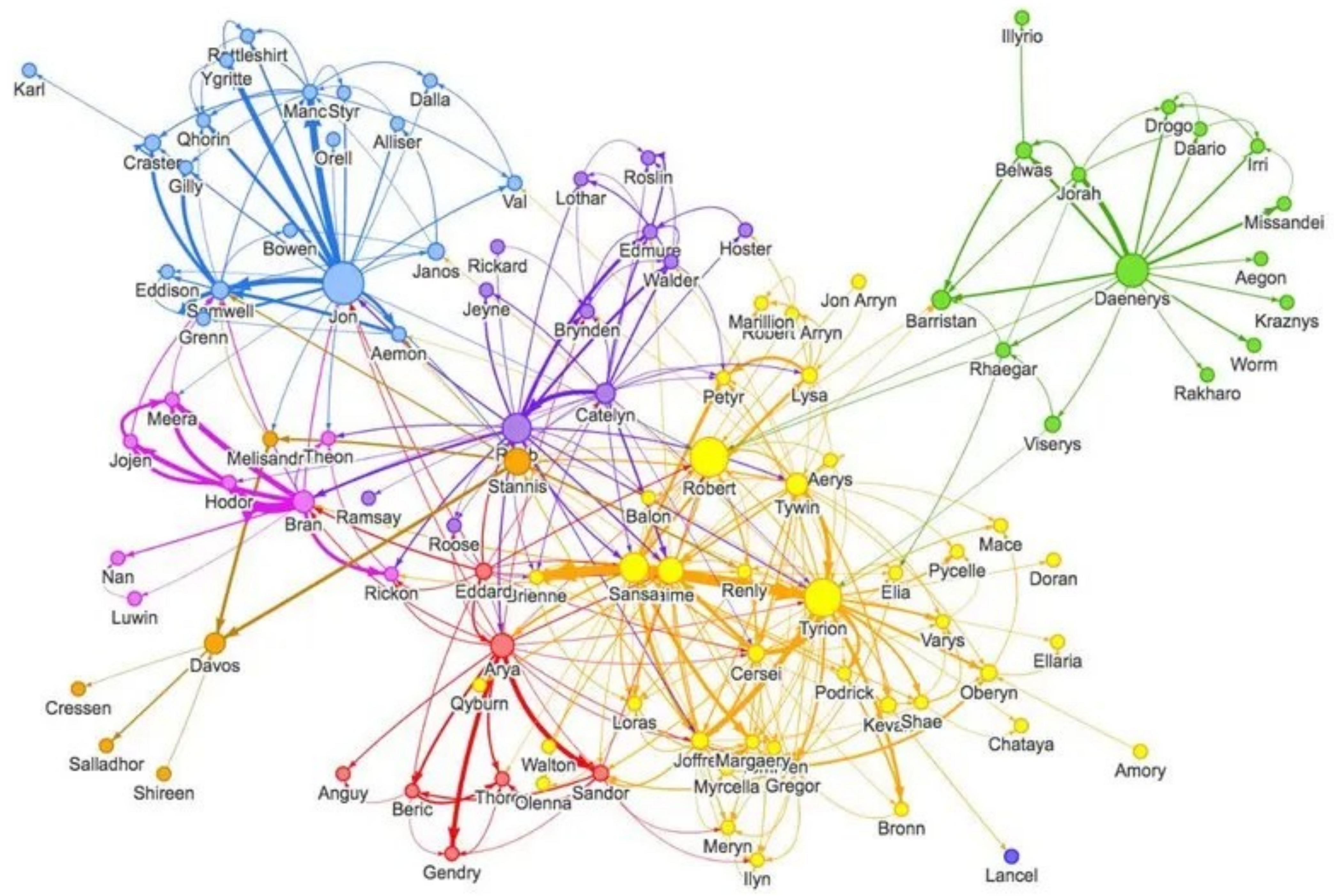


https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png
https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

Spectral/Algebraic Graph Theory

Graphs can be viewed as matrices.

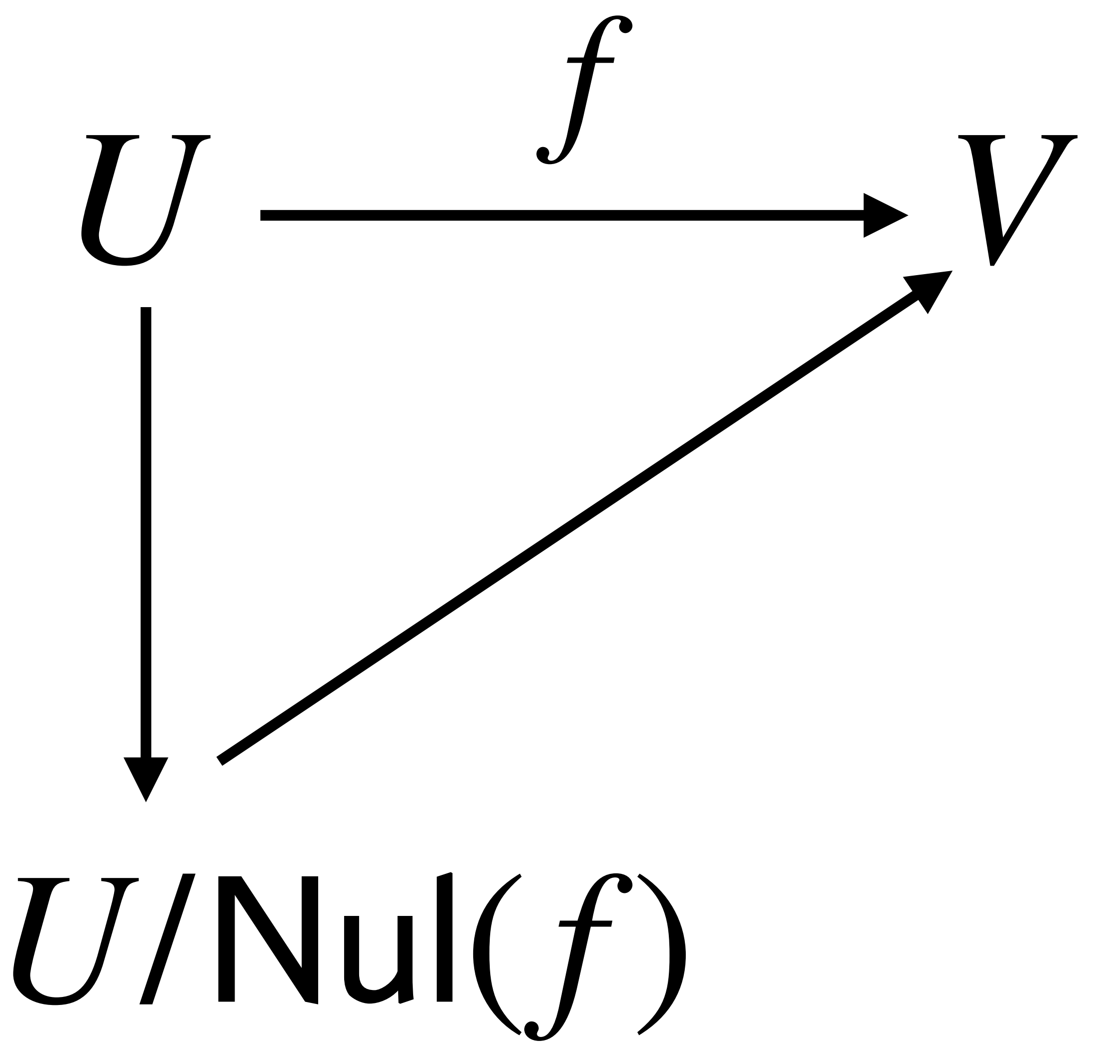
The finding eigenvalues in graphs can gives use better clustering and cutting algorithms.



https://medium.com/@n.rajadhyaksha/a-gentle-introduction-to-graph-spectral-filtering-df03ddc9d3f7

Abstract Algebra

$$\frac{U}{\mathrm{Nul}(f)} \cong \mathrm{Range}(f)$$



There's a lot of beautiful structure in the algebra we've done in this course.

And there are lots of directions to go from here (infinite dimensional spaces, less restrictive settings like groups and modules,...)

Course List

- CS 365 Foundations of Data Science
- CS 440 Intro to Artificial Intelligence
- CS 480 Intro to Computer Graphics
- CS 505 Intro to Natural Language Processing
- CS 506 Tools for Data Science
- CS 507 Intro to Optimization in ML
- CS 523 Deep Learning
- CS 530 Advanced Algorithms
- CS 531 Advanced Optimization Algorithms
- CS 542 Machine Learning
- CS 565 Algorithmic Data Mining
- CS 581 Computational Fabrication
- CS 583 Audio Computation

Some of these may not exist anymore...

Appreciations

The Course Staff

I'd like to thank:

Abhinit Sati, Vishesh Jain, Ieva Sagaitis, Kevin Wrenn, Jin Zhang, Sohan Atluri, Fynn Buesnel, Aseef Imran, Eugene Jung, Chris Min, Wyatt Napier, Kyle Yung

If you see them around you should thank them as well

The CS Department Staff

```
If you're ever in the CS Department office, be
kind to the people who work there. They work
very hard to keep all our courses running
```

The Students of CS132

```
Thanks for sticking with it

Thanks for giving feedback

Thanks for participating
```

fin