Eigen and Singular Value Decomposition

Contents

1 Eigenvalue Decomposition **3**
- 1.1 Intuition . . . 3
- 1.2 Applications . . . 3
- 1.3 Definition . . . 3
- 1.4 Linear Systems and Eigenvalue Decomposition . . . 4
- 1.5 Diagonalization . . . 4
- 1.6 The Spectral Theorem . . . 6
- 1.7 Example . . . 8
- 1.8 Questions . . . 8

2 Singular Value Decomposition **10**
- 2.1 Definition . . . 10
- 2.2 Geometric Interpretation . . . 11
- 2.3 Intuitive Explanation . . . 11
- 2.4 Reduced and Full SVD . . . 12
- 2.5 Key Properties . . . 13
  - 2.5.1 Existence and Uniqueness . . . 13
  - 2.5.2 Relation to Eigenvalue Decomposition . . . 15
- 2.6 Questions . . . 16

3 Ranking from an Adjacency Matrix **17**
- 3.1 Adjacency Matrix and Influence Propagation . . . 17
- 3.2 From Degree Ranking to Eigenvector-Based Ranking . . . 17
- 3.3 Eigenvector-Based Ranking . . . 18
- 3.4 Power Iteration for Ranking . . . 18
- 3.5 Example . . . 19
- 3.6 From Raw Adjacency to Stochastic Matrices . . . 19
- 3.7 Normalized Variants . . . 19
  - 3.7.1 PageRank . . . 19
  - 3.7.2 Laplacian Normalization . . . 20
- 3.8 Ranking nodes using SVD . . . 20
- 3.9 Low-Rank Approximations . . . 21
- 3.10 Questions . . . 22

4 Linear Algebra Perspective of Principal Component Analysis 24
4.1 Intuitive Explanation 24
4.2 Covariance Matrices and Variance Maximization 24
4.3 Formal PCA via Spectral Decomposition 25
4.4 PCA and the Dual Formulation: $XX^\top$ 25
4.5 Connection with Singular Value Decomposition 25
4.6 Feature Mapping and Dimensionality Reduction 26
4.7 Questions 26

5 Computing Eigenvalues and Eigenvectors: Numerical Methods (Bonus) 28
5.1 The Characteristic Polynomial 28
5.2 The Rayleigh Quotient 29
5.3 Power Iteration 29
5.4 Inverse Iteration and the Shifted Power Method 30
5.5 QR Algorithm and Schur Decomposition 32
5.6 Conclusion: Choosing the Right Eigenvalue Method 34
5.7 Questions 36

6 Practical Session Question 37

1 Eigenvalue Decomposition

1.1 Intuition

The **eigenvalue decomposition** allows to analyze and simplify a linear transformation by expressing it in a coordinate system where it becomes diagonal. Given a square and full-rank matrix, its decomposition outputs the so-called eigenvectors and their respective eigenvalues. The eigenvectors represent the **principal directions** in which the matrix acts in a simple way: along each eigenvector, the matrix only multiplies vectors by a scalar, the corresponding eigenvalue. Therefore, by switching to the eigenvector basis, we turn a complicated transformation into a simple diagonal one.

1.2 Applications

Examples of major applications based on Eigenvalue decomposition include:

- **Differential equations**: solving systems $\dot{x} = Ax$.
- **Stability analysis**: sign of eigenvalues $\Rightarrow$ growth or decay.
- **Network analysis**: steady-state distribution analysis in Markov chains. Ranking problems in networks.

1.3 Definition

Let $A \in \mathbb{R}^{n\times n}$ (or $\mathbb{C}^{n\times n}$) be a **square**, **diagonalizable** matrix. Then there exists an invertible matrix $X \in \mathbb{R}^{n\times n}$ and a diagonal matrix $D \in \mathbb{R}^{n\times n}$ such that

$$A = XDX^{-1}$$

where the diagonal entries of D are the **eigenvalues** $\lambda_1, \ldots, \lambda_n$ and the columns of X are the **eigenvectors** of A such that:

$$Ax_i = \lambda_i x_i,$$

Thus, for any vector x,

$$Ax = XDX^{-1}x.$$

This representation means that in an appropriate basis (the eigenvector basis), the matrix A acts simply by *stretching or contracting* each coordinate by its corresponding eigenvalue. Indeed, by applying X^{-1} to x we map x to the *eigenvector basis*, then by applying D to $X^{-1}x$ we simply *stretch or contract each coordinate* by its corresponding eigenvalue, and finally by applying X to $DX^{-1}x$ we simply *map back* to the resulting vector to the original coordinate system.

1.4 Linear Systems and Eigenvalue Decomposition

To see why Eigenvalue Decomposition is useful, consider rewriting the system $Ax = b$ in the eigenvector basis. Define:

$$\hat{x} = X^{-1}x, \qquad \hat{b} = X^{-1}b.$$

Applying the decomposition:

$$Ax = b \quad \Longleftrightarrow \quad XDX^{-1}x = b \quad \Longleftrightarrow \quad D\hat{x} = \hat{b}.$$

Thus, instead of solving a full $n \times n$ system, we solve n **independent scalar equations**:

$$\lambda_i \hat{x}_i = \hat{b}_i, \qquad i = 1, \ldots, n.$$

1.5 Diagonalization

Two important conditions should be fulfilled to apply this decomposition: i) A must be **square**. ii) A must be **diagonalizable**: it should have n linearly independent eigenvectors to form a basis of $\mathbb{R}^n$.

Algebraic and Geometric Multiplicity are two important criteria used to determine if a matrix is diagonalizable. For each eigenvalue λ of A:

- The **algebraic multiplicity** (AM) of λ is the number of times λ appears as a root of the characteristic polynomial:

  $$p_A(x) = \det(xI - A).$$

  Intuitively the algebraic multiplicity counts *how many independent eigenvectors this eigenvalue should have.*

- The **geometric multiplicity** (GM) of λ is the dimension of its eigenspace:

  $$\mathrm{GM}(\lambda) = \dim \ker(A - \lambda I).$$

  Intuitively the geometric multiplicity counts *how many independent eigenvectors actually exist for that eigenvalue*

A matrix is diagonalizable if and only if, for every eigenvalue λ,

$$\mathrm{GM}(\lambda) = \mathrm{AM}(\lambda).$$

If $\mathrm{GM}(\lambda) < \mathrm{AM}(\lambda)$, the matrix does not have enough independent eigenvectors; it is called **defective** and cannot be diagonalized.

Practical Criteria Several equivalent conditions, that are sometimes simpler and more useful in real-world cases, allow us to determine whether a matrix is diagonalizable:

1. **If all eigenvalues are distinct, the matrix is diagonalizable.**
   Distinct eigenvalues automatically give one eigenvector per eigenvalue. In this case both the geometric and arithmetic multiplicities are equal to one. This condition is sufficient but not necessary.

2. **Symmetric matrices are always diagonalizable.**
   If $A = A^T$ (or $A = A^*$ in the complex case), then

   $$A = Q\Lambda Q^T,$$

   with Q orthogonal and Λ diagonal (this property derives from the so-called Spectral Theorem, and it is the foundation of PCA, and many other techniques).

3. **Rank condition:**
   Let X be the matrix of eigenvectors **returned numerically**, then A is diagonalizable if:

   $$\text{rank}(X) = n,$$

Geometric intuition: why some matrices are not diagonalizable? Diagonalization requires that the action of A stretches/contracts the space along n independent directions. When a matrix "collapses" multiple directions into the same eigenspace, we lose these degrees of freedom and diagonalization becomes impossible.

A classical example is the shear matrix

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Let us compute the characteristic polynomial $p_A(x) = det(xI - A) = (x-1)^2$. Hence A has only one eigenvalue $\lambda = 1$ with algebraic multiplicity $AM = 2$. In order to determine the geometric multiplicity let us solve $Ax = x$:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\begin{pmatrix} x_1 + x_2 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Therefore $x_2 = 0$ and consequently only one independent eigenvector exists. The geometric multiplicity is then $GM = 1$. $GM \neq AM$ for $\lambda = 1$ so A cannot be diagonalized.

1.6 The Spectral Theorem

Intuitively, a symmetric matrix acts like a system that *does not twist the space*, but *only stretches it* along special orthogonal directions. While a general matrix may shear, rotate, and mix dimensions, a symmetric matrix behaves in a fundamentally simpler way: 1) it has real eigenvalues, 2) its eigenvectors corresponding to different eigenvalues are orthogonal, 3) it can be "aligned" with a set of perpendicular axes.

Geometrically, applying a symmetric matrix is equivalent to:

$$\text{rotate coordinates} \;\rightarrow\; \text{stretch independently} \;\rightarrow\; \text{rotate back.}$$

This is exactly the decomposition

$$A = Q\Lambda Q^{\top},$$

where Q is an orthogonal (or unitary) matrix and Λ is diagonal.

Theorem 1 (Spectral Theorem)**.** *Let $A \in \mathbb{R}^{n\times n}$ be a matrix. If A is real symmetric ($A = A^T$), then A has n real eigenvalues and admits an orthonormal (unitary) basis of eigenvectors. (The same holds if A is complex Hermitian, i.e., $A = A^*$), and then A can be written as:*

$$A = Q\Lambda Q^{*},$$

where Q is orthogonal (or unitary) and Λ is diagonal with the eigenvalues of A on the diagonal.

Proof

Hermitian matrices have real eigenvalues. Let v be an eigenvector of A with eigenvalue λ:

$$Av = \lambda v.$$

Taking the inner product with v, and using $A = A^*$ (Hermitian matrix definition):

$$v^* Av = v^*(\lambda v) = \lambda\, v^* v.$$

But also

$$v^* Av = (Av)^* v = (\lambda v)^* v = \lambda^*\, v^* v.$$

Thus,

$$\lambda v^* v = \lambda^* v^* v.$$

Since $v^* v > 0$, we obtain $\lambda = \lambda^*$, so λ is real.

Eigenvectors corresponding to distinct eigenvalues are orthogonal. Let $Av_1 = \lambda_1 v_1$ and $Av_2 = \lambda_2 v_2$ with $\lambda_1 \neq \lambda_2$. Compute:

$$v_1^* A v_2 = \lambda_2 \, v_1^* v_2.$$

But since $A = A^*$:

$$v_1^* A v_2 = (Av_1)^* v_2 = (\lambda_1 v_1)^* v_2 = \lambda_1 \, v_1^* v_2.$$

Thus,

$$\lambda_1 \, v_1^* v_2 = \lambda_2 \, v_1^* v_2.$$

Since $\lambda_1 \neq \lambda_2$, the only possibility is

$$v_1^* v_2 = 0.$$

Construction of an orthonormal eigenbasis. Since A is Hermitian, it always has at least one (real) eigenvalue and an associated eigenvector $v_1 \neq 0$.

Let us consider V_1 the orthogonal complement of v_1, i.e., the $n-1$ dimensional hyperplane consisting of all vectors perpendicular to v_1:

$$V_1 = \{x \in \mathbb{K}^n : v_1^* x = 0\}.$$

If $x \in V_1$, then $Ax \in V_1$ (i.e., V_1 is invariant under A):

$$v_1^*(Ax) = (A^* v_1)^* x = (Av_1)^* x = (\lambda_1 v_1)^* x = \lambda_1 v_1^* x = 0.$$

This property *depends crucially on symmetry*: for a nonsymmetric matrix this is not true in general, which is why the spectral theorem does not hold for arbitrary matrices. Let us restrict A to V_1 and apply the same argument again. The restriction $A|_{V_1}$ is itself Hermitian (the Hermitian property is preserved under restriction to an invariant subspace). Therefore it also has an eigenvalue and an eigenvector, call it v_2, and v_2 lies *entirely* inside V_1.

Since v_2 is in the orthogonal complement of v_1, we automatically have

$$v_1^* v_2 = 0.$$

By repeating recursively, at the k-th step, we have already found k mutually orthogonal eigenvectors $v_1, \ldots, v_k$. Their orthogonal complement

$$V_k = \{x \in \mathbb{K}^n : v_i^* x = 0 \text{ for } i = 1, \ldots, k\}$$

has dimension $n-k$ and is invariant under A. Restricting A to V_k, we again obtain an eigenvalue and an eigenvector v_{k+1} inside V_k, which is therefore orthogonal to all previous eigenvectors.

After n steps, we obtain n mutually orthogonal eigenvectors. Dividing each by its norm yields an orthonormal eigenbasis. Let Q be the matrix whose columns are these orthonormal eigenvectors, and let Λ be the diagonal matrix of the corresponding eigenvalues. Then

$$A = Q \Lambda Q^*.$$

1.7 Example

Let

$$A = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix}.$$

The characteristic polynomial is $p_A(x) = det(xI - A) = (x-3)(x-2)$ The eigenvalues are $\lambda_1 = 3$, $\lambda_2 = 2$. By solving $Ax = 3x$ and $Ax = 2x$ we find that the corresponding eigenvectors are:

$$x_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad x_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Thus,

$$X = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}, \qquad D = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}.$$

Solving $Ax = b$ becomes:

$$D\hat{x} = \hat{b}, \qquad \text{with } \hat{x} = X^{-1}x,\ \hat{b} = X^{-1}b.$$

Each coordinate solves:

$$3\hat{x}_1 = \hat{b}_1, \qquad 2\hat{x}_2 = \hat{b}_2.$$

Summary:

- The matrix should be square and diagonalizable.
- The matrix is diagonalizable if it is symetric, it has n distinct eigenvalues or the matrix of eigenvectors returned numerically is full-rank.
- Eigenvectors $\Rightarrow$ directions in space left unchanged by A.
- Eigenvalues $\Rightarrow$ factor by which the matrix stretches or contracts those directions.
- The Eigenvalue Decomposition of a $n \times n$ system transforms it into n independent 1D problems.

1.8 Questions

1. Compute the eigenvalues and eigenvectors of $A = \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix}$.
2. For the matrix $B = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$, explain why it is diagonalizable.
3. Give the definition of **algebraic** and **geometric** multiplicity. Provide an example of a matrix where these multiplicities differ.

4. Explain why a matrix with a full set of linearly independent eigenvectors is diagonalizable.

5. Let A be diagonalizable. Show that $A^k = X\Lambda^k X^{-1}$ and explain why this is useful computationally.

6. Explain in simple terms the geometric meaning of an eigenvector.

7. If a matrix has eigenvalues $\lambda_1 = 1$, $\lambda_2 = 1$, is it necessarily diagonalizable? Justify.

8. If you want to determine long-term behavior of a linear dynamical system $x_{k+1} = Ax_k$, which eigenvalue(s) matter most? Why?

9. Explain why eigenvalues of a triangular matrix are its diagonal entries.

10. A biologist provides you with a non-square matrix and asks for its "eigenvalue decomposition." What do you answer?

2 Singular Value Decomposition

2.1 Definition

The **Singular Value Decomposition (SVD)** is a major matrix factorization in linear algebra. Unlike the **Eigenvalue Decomposition**, the SVD exists for *any* real matrix $A \in \mathbb{R}^{m \times n}$ and it can be written as:

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

$$\underbrace{A}_{A \in \mathbb{R}^{m \times n}} = \underbrace{\begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_m \\ | & | & & | \end{bmatrix}}_{U \in \mathbb{R}^{m \times m}} \underbrace{\begin{bmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \\ 0 & & & \end{bmatrix}}_{\Sigma \in \mathbb{R}^{m \times n}} \underbrace{\begin{bmatrix} | & | & & | \\ v_1^\top & v_2^\top & \cdots & v_n^\top \\ | & | & & | \end{bmatrix}}_{V^\top \in \mathbb{R}^{n \times n}}$$

Left and right singular vectors

U is an $m \times m$ **unitary matrix**, and its columns are the **left-singular vectors**. V is an $n \times n$ **unitary matrix**, and its columns are the **right-singular vectors**. The singular vectors describe the directions along which these stretchings occur. A square matrix Q is **unitary** if its columns form an orthonormal basis:

$$Q^\tau Q = Q Q^\tau = I,$$

where Q^τ is the conjugate transpose of Q (for real matrices $Q^* = Q^\top$). In other words, the different columns of a unitary matrix Q are normed, and they are orthogonal to each other. Interestingly, unitary matrices preserve lengths and inner products: $\|Qx\|_2 = \|x\|_2$ and $\langle Qx, Qy \rangle = \langle x, y \rangle$

Singular values

Σ is a $m \times n$ **rectangular diagonal matrix**: it is not square, but all its nonzero entries lie on the main diagonal, and all off-diagonal entries are zero.

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0, \qquad p = \min(m, n).$$

The singular values measure how strongly the matrix A stretches space in specific directions.

Alternative notation

A non-negative real number σ is called a *singular value* of a matrix $A \in \mathbb{R}^{m\times n}$ if and only if there exist unit vectors

$$u \in \mathbb{R}^m, \qquad v \in \mathbb{R}^n$$

such that

$$Av = \sigma\, u, \qquad A^\top u = \sigma\, v.$$

The vector u is a *left-singular vector* and v is a *right-singular vector* associated with the singular value σ.

Complex matrices

The SVD also exists for *any* complex matrix $A \in \mathbb{C}^{m\times n}$. In this case, the conjugate transposes are considered instead of the transposes (e.g. we condider V^* the conjugate transpose of V instead of its transpose $V^\top$).

2.2 Geometric Interpretation

Consider the **unit sphere** in $\mathbb{R}^n$:

$$S = \{x \in \mathbb{R}^n \,:\, \|x\| = 1\}.$$

Applying A to this sphere produces a **hyper-ellipse** in $\mathbb{R}^m$. The decomposition $A = U\Sigma V^\top$ explains this transformation in three stages:

1. $V^\top$: rotates the sphere but preserves its shape since V is unitary.
2. Σ: stretches the rotated sphere along orthogonal axes with lengths $\sigma_1, \ldots, \sigma_p$.
3. U: rotates the resulting ellipse in the output space.

Thus, the vectors $\{u_1, \ldots, u_m\}$ give the **principal axis directions** of the output ellipse, the singular values $\{\sigma_1, \ldots, \sigma_m\}$ give the length of the **semi-axis lengths** of the ellipse and the vectors $\sigma_i u_i$ represent the semi-principal axes of the hyper-ellipse. If Rank $A = r$, then exactly r singular values are non-zero. If $m \geq n$, then at most n axes of the ellipsoid can have non-zero length.

$$\text{circle} \xrightarrow{\text{Rotation}} \text{axis-aligned circle} \xrightarrow{\text{stretch}} \text{axis-aligned ellipse} \xrightarrow{\text{rotation}} \text{output ellipse.}$$

2.3 Intuitive Explanation

The SVD answers the question:

In what directions does a matrix stretch space, and by how much?

Every matrix acts like:

$$\text{rotation} \quad \rightarrow \quad \text{scaling along orthogonal directions} \quad \rightarrow \quad \text{rotation.}$$

The SVD simply reveals:

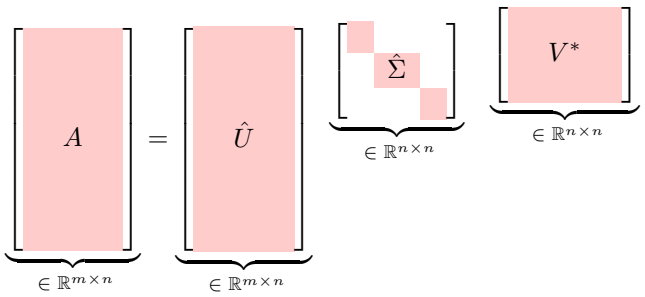
- The directions that are maximally amplified (v_1, v_2, ...),
- The amount of amplification (σ_1, σ_2, ...)
- The resulting output directions (u_1, u_2, ...).

2.4 Reduced and Full SVD

Reduced SVD

Assume $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ with Rank $A = n \leq m$. Then the SVD may be written in its **reduced form** that is computationally cheaper and often used in practice:

$$A = \hat{U}\hat{\Sigma}V^\top$$



- $\hat{U}$ is $m \times n$, with orthonormal columns (left singular vectors).
- V is $n \times n$, orthonormal (right singular vectors).
- $\hat{\Sigma}$ is $n \times n$ diagonal and contains only the non-zero singular values.

Notice that if $m > n$, $\hat{U}$ does not span all of $\mathbb{R}^m$, and $\hat{U}$ does not define a basis. Moreover, if Rank $A = r$ and $r < min(m, n)$, then: A has only r non-zero singular values, and can be written, in its reduced form as:

$$A = \hat{U}\hat{\Sigma}\hat{V}^\top$$

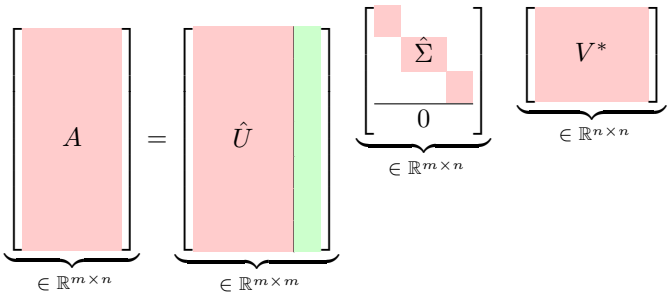
With $\hat{\Sigma} \in \mathbb{R}^{(r\times r)}$ a diagonal matrix containing the r singular values, $\hat{U} \in \mathbb{R}^{(m\times r)}$ and $\hat{V} \in \mathbb{R}^{(n\times r)}$ having both r left and right singular vectors associated.

Full SVD

To obtain a full unitary matrix U, as defined in the Section 2.1, we need to: i) Add $m-n$ orthonormal vectors to complete the basis and, ii) append $m-n$ zero rows to $\hat{\Sigma}$ to compensate.

Then

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



with $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ fully unitary.

2.5 Key Properties

2.5.1 Existence and Uniqueness

Hereafter we provide properties and sketches of proofs for complex matrices, notice that the same applies to real ones.

1. Every matrix $A \in \mathbb{C}^{m\times n}$ admits a singular value decomposition $A = U\Sigma V^*$.
2. The singular values $\{\sigma_i\}$ are uniquely determined (up to ordering).
3. If the nonzero singular values are distinct, the corresponding singular vectors u_i and v_i are unique up to a complex phase factor.

Existence. Consider the Hermitian positive semidefinite matrix $A^*A \in \mathbb{C}^{n\times n}$. By the spectral theorem for Hermitian matrices there exists a unitary matrix V and a real diagonal matrix $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ with $\lambda_i \geq 0$ such that

$$A^*A = V\Lambda V^*,$$

and such that the columns $v_1, \ldots, v_n$ of V form an orthonormal basis of $\mathbb{C}^n$. Let us define the nonnegative numbers

$$\sigma_i = \sqrt{\lambda_i} \geq 0, \qquad i = 1, \ldots, n.$$

Let us order them so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ with $p = \min(m, n)$. And let us partition the diagonal matrix as $\Lambda = \Sigma^2$ where $\Sigma \in \mathbb{R}^{n\times n}$ is the diagonal matrix whose first p diagonal entries are the σ_i. Then:

$$A^*A = V\Sigma^2 V^*.$$

For each index i with $\sigma_i > 0$ define

$$u_i \;=\; \frac{1}{\sigma_i}\, Av_i \in \mathbb{C}^m.$$

Vectors $\{u_i\}$ are normed, since:

$$\|u_i\|_2 = \frac{1}{\sigma_i}\|Av_i\|_2 = \frac{1}{\sigma_i}\sqrt{v_i^* A^* A v_i} = \frac{1}{\sigma_i}\sqrt{v_i^*(\sigma_i^2 v_i)} = 1,$$

and all pairs of vectors u_i and u_j (such that $i \neq j$), corresponding to nonzero singular values, are orthogonal to each other:

$$u_i^* u_j = \frac{1}{\sigma_i\sigma_j} v_i^* A^* A v_j = \frac{1}{\sigma_i\sigma_j} v_i^*(\sigma_j^2 v_j) = 0,$$

Collect the left singular vectors u_i (for $\sigma_i > 0$) as columns of a matrix $\hat{U}$, and collect the corresponding right singular vectors v_i as columns of $\hat{V}$. One then has for those indices

$$Av_i = \sigma_i u_i, \qquad A^* u_i = \sigma_i v_i.$$

From these relations we obtain

$$A\hat{V} = \hat{U}\hat{\Sigma},$$

where $\hat{\Sigma}$ is the diagonal matrix of the nonzero singular values. If $m > n$ or there are zero singular values, extend $\hat{U}$ to a full orthonormal basis of $\mathbb{C}^m$ (by adding orthonormal vectors spanning the orthogonal complement of the columns of $\hat{U}$), and extend $\hat{V}$ similarly if needed. Appending zero rows/columns to make sizes match yields the full factorization

$$A = U\Sigma V^*.$$

Uniqueness of the singular values. The singular values σ_i are by construction the nonnegative square roots of the eigenvalues of A^*A. Since the eigenvalues of A^*A are uniquely determined (counting multiplicity), the multiset $\{\sigma_i\}$ is uniquely determined (up to ordering). Hence the singular values are unique.

Uniqueness of singular vectors when singular values are distinct. Consider an index i with $\sigma_i > 0$. The right singular vector v_i satisfies

$$A^* A v_i = \sigma_i^2 v_i,$$

so v_i is an eigenvector of A^*A associated with the eigenvalue σ_i^2. If σ_i^2 is a simple eigenvalue (algebraic multiplicity 1), then its eigenspace is one-dimensional, so any eigenvector is unique up to multiplication by a nonzero scalar. Requiring $\|v_i\| = 1$ restricts that scalar to a complex number of unit modulus (a "phase"),

so v_i is unique up to multiplication by $e^{i\theta}$. The corresponding left singular vector is then

$$u_i = \frac{1}{\sigma_i} A v_i,$$

which inherits the same phase freedom; hence u_i is unique up to the same complex phase. Thus when the singular values are distinct, the associated singular vectors are determined uniquely up to multiplication by scalars of unit modulus.

Multiplicity and Degeneracy. A singular value σ is called *degenerate* if it has more than one linearly independent left- (or right-) singular vector, i.e., if it has multiplicity $k > 1$. Its corresponding eigenspace is k-dimensional and the right singular vectors are not unique: any normalized linear combination of vectors in such a singular subspace is again a singular vector associated with σ. If a singular value is *non-degenerate*, its singular vectors are unique up to multiplication by a unimodular scalar (a phase factor or a sign in the real case). When all singular values are non-zero and non-degenerate, the SVD is unique up to joint multiplication of the ith columns of U and V by the same unimodular scalar. In the general case, the SVD is unique only up to unitary transformations inside each singular subspace.

Singular Vectors. Let us consider a matrix $A \in \mathbb{C}^{(n\times m)}$, and let $p = \min(m, n)$. The first p columns of U and V contain the left- and right-singular vectors corresponding to these singular values. From this, we deduce that A has at most p distinct singular values. There always exists a unitary basis U of $\mathcal{C}^{(m\times m)}$ whose columns include left-singular vectors of each singular value. There always exists a unitary basis V of $\mathcal{C}^{(n\times n)}$ whose columns include right-singular vectors of each singular value. As aforementioned, if $m > n$, then U must include $m - n$ additional orthonormal vectors to form a basis in $\mathbb{C}^{(m\times m)}$. While if $m < n$, then V must include $n - m$ additional orthonormal vectors to form a basis in $\mathbb{C}^{(n\times n)}$.

2.5.2 Relation to Eigenvalue Decomposition

SVD is closely related to the eigenvalue decompositions of the Hermitian matrices A^*A and AA^*. Indeed, given an SVD $A = U\Sigma V^*$, we have

$$A^*A = V(\Sigma^*\Sigma)V^*, \qquad AA^* = U(\Sigma\Sigma^*)U^*.$$

Consequently, the columns of V are eigenvectors of A^*A, the columns of U are eigenvectors of AA^*, and the non-zero singular values satisfy

$$\sigma_i = \sqrt{\lambda_i(M^*M)} = \sqrt{\lambda_i(MM^*)}.$$

Despite the relationship between Eigenvalue Decomposition and SVD, you should remember that Eigenvalue decomposition exists only for diagonalizable square matrices while singular value decomposition exists for **any** $m \times n$ matrix. Eigenvalues may be complex or negative; singular values are always real and non-negative. Eigenvalue decomposition uses one matrix of eigenvectors; the SVD

uses two unitary matrices U and V. The SVD always produces orthonormal bases aligned with the geometry of A, even when A is non-normal.

Summary:

- The Singular Value Decomposition (SVD) applies to *any* matrix, square or rectangular.
- Singular values are always real and non-negative; they are the square roots of the eigenvalues of A^*A or AA^*.
- Right-singular vectors (columns of V) are eigenvectors of A^*A; Left-singular vectors (columns of U) are eigenvectors of AA^*.
- Each singular value σ_i measures how much A stretches vectors along the corresponding singular direction.
- The SVD expresses A as $A = U\Sigma V^*$, separating rotation, scaling, and another rotation.

2.6 Questions

1. Compute the SVD of $A = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$.
2. Explain why the SVD exists for *every* matrix, unlike eigenvalue decomposition.
3. Give the geometric interpretation of the SVD as transforming the unit sphere into an ellipsoid.
4. Explain the difference between full SVD and reduced SVD.
5. Describe how singular values relate to the eigenvalues of A^TA.
6. In data compression (e.g. images), why does keeping only the largest singular values often work well?
7. How can you compute the rank of the matrix using its singular values.
8. Explain the meaning of left singular vectors vs right singular vectors.
9. You are given noisy biological data in a matrix. Explain why SVD-based denoising might help.
10. A colleague wants to use SVD to infer causal relationships in genetics data. What do you think about this?

3 Ranking from an Adjacency Matrix

Ranking nodes from an adjacency matrix is an important task in network analysis. The goal is to determine the "importance" of each node based on the structure of the graph. This idea underlies many applications such as webpage ranking created by Google (PageRank Algorithm), analyzing social networks of scientific citation networks, and building recommender systems.

3.1 Adjacency Matrix and Influence Propagation

Given a directed graph with n nodes, its adjacency matrix $A \in \mathbb{R}^{n \times n}$ is defined as:

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge from node } j \text{ to node } i, \\ 0 & \text{otherwise.} \end{cases}$$

Each column represents outgoing edges of a node, and each row represents incoming edges.

3.2 From Degree Ranking to Eigenvector-Based Ranking

A first, very simple way to rank nodes in a graph is to use the degree of each node. For a directed graph, one may define the *in-degree* of node i as the number of incoming edges:

$$d_i = \sum_{j=1}^{n} A_{ij}.$$

Nodes with high in-degree receive many connections and can be considered more "important". This measure is intuitive and computationally trivial, but it treats all incoming edges equally, regardless of the importance of the nodes providing these edges. Notice that depending on the application, one could rely on the out-degree instead.

Weighted Degree Ranking. To refine this idea, one can weight each incoming edge by the importance of the node it comes from. In other words, **A node is important if it is pointed to by other important nodes.** Thus, a node linked from influential nodes should itself become more influential. If x_j denotes the importance of node j, then the score of node i becomes:

$$x_i \propto \sum_{j=1}^{n} A_{ij}\, x_j.$$

This recursive formula states that the importance of a node is proportional to the sum of the importance of its neighbors pointing to it. In vector notation:

$$x \propto Ax.$$

Final Normalized Eigenvector Ranking. Since the trivial solution $x = 0$ is uninformative, we seek a nonzero vector satisfying

$$Ax = \lambda x,$$

i.e., an eigenvector associated with the dominant eigenvalue λ. To make this a proper ranking, we normalize the vector:

$$x \leftarrow \frac{x}{\|x\|_1},$$

which yields a probability-like importance distribution. This leads directly to the classical *eigenvector centrality* and forms the basis for PageRank and other modern ranking algorithms.

3.3 Eigenvector-Based Ranking

A natural idea is to define a ranking vector $x \in \mathbb{R}^n$ such that

$$\lambda x = Ax,$$

i.e., importance is proportional to the sum of importances of linking neighbors.

This equation is equivalent to the dominant eigenvector problem:

$$Ax = \lambda x,$$

where the eigenvector associated with the largest eigenvalue λ gives relative ranking scores. For graphs with positive connections, **Perron–Frobenius theorem** guarantees:

- a unique positive dominant eigenvalue,
- a unique eigenvector with strictly positive entries,
- convergence of power iteration.

3.4 Power Iteration for Ranking

The most common algorithm for computing the eigenvector-based ranking is the **power iteration** method:

1. Start with x_0 (e.g., uniform vector of ones).
2. Iteratively compute:
$$x_{k+1} = \frac{Ax_k}{\|Ax_k\|_1}.$$
3. Continue until convergence.

This method is extremely efficient, especially for large sparse graphs.

3.5 Example

Consider the graph:

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Interpretation:

- Node 1 receives edges from nodes 2 and 3.
- Node 2 receives an edge from node 1.
- Node 3 receives an edge from node 2.

A few iterations of power method yield:

$$x = (0.743, 0.557, 0.371)^T,$$

indicating node 1 is the most central: it is referenced by two nodes, including a node that is itself influential.

3.6 From Raw Adjacency to Stochastic Matrices

For ranking, we typically convert the adjacency matrix into a column-stochastic transition matrix:

$$P_{ij} = \frac{A_{ij}}{\sum_k A_{kj}}.$$

This ensures that each node distributes its importance evenly among its outgoing neighbors.

Interpretation. The ranking vector x becomes the stationary distribution of a random walk on the graph:

$$x = Px.$$

Thus, ranking corresponds to the long-term visiting frequency of a random surfer.

3.7 Normalized Variants

3.7.1 PageRank

The classical PageRank variant modifies the stochastic matrix to avoid dead ends and cycles:

$$P' = \alpha P + (1 - \alpha) \left(\frac{1}{n} \mathbf{1}\mathbf{1}^T \right),$$

where $\alpha \in [0.85, 0.95]$ is a damping parameter.

The ranking vector is the principal eigenvector of P':

$$x = P'x.$$

Intuition. With probability α, a random walker follows links. With probability $1 - \alpha$, the walker "teleports" to a random node. This prevents pathological structures and ensures a unique positive eigenvector for all graphs.

3.7.2 Laplacian Normalization

Let the *out-degree matrix* D of A be the diagonal matrix defined by $D_{ii} = \sum_{j=1}^{n} A_{ij}$ (we could respectively define the *in-degree matrix* as $D_{ii} = \sum_{i=1}^{n} A_{ij}$). i.e. each diagonal entry stores the total weight of edges leaving node i. The (unnormalized) graph Laplacian is then: L = D - A.

In practice, it is often preferable to work with a *normalized Laplacian*, which compensates for heterogeneous node degrees and prevents high-degree nodes from dominating the analysis. The most common choice is the *symmetric normalized Laplacian*: $L_{\text{sym}} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}$

To compute L_{sym}, one first builds A, then computes the degree matrix D, forms $D^{-1/2}$ by taking the inverse square root of each diagonal entry ($D_{ii}^{-1/2} = 1/\sqrt{D_{ii}}$, assuming $D_{ii} > 0$), and finally applies the similarity transformation above. This normalization rescales edge weights by the degrees of the incident nodes, so that each edge contributes proportionally rather than absolutely. As a result, the spectrum of L_{sym} reflects the *relative* connectivity structure of the graph. In some cases, the matrices are made symetric before computing the Laplacian: $A_{sym} = \frac{1}{2}(A + A^\top)$

3.8 Ranking nodes using SVD

Considering the adjacency matrix A (or a normalized version), it is also possible to use its singular value decomposition of A to rank nodes. The SVD of A is given by $A = U \Sigma V^\top$, where, $U \in \mathbb{R}^{n \times n}$ contains the left singular vectors, $V \in \mathbb{R}^{n \times n}$ contains the right singular vectors and $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$, and each singular value σ_k represents the strength of a kind of "latent connectivity pattern" in the network.

Undirected Networks If the network is undirected, the adjacency matrix A is symmetric and $U = V$ In this case, the leading singular vector u_1 (associated with σ_1) provides a natural node ranking. The importance score of node i is defined as $\text{score}(i) = |u_1(i)|$ This ranking coincides with *eigenvector centrality*, which assigns high scores to nodes connected to other important nodes.

Directed Networks In directed networks, the adjacency matrix A is generally asymmetric.

As a result, rows and columns represent fundamentally different types of information: Rows correspond to *outgoing* connectivity patterns, while columns correspond to *incoming* connectivity patterns. The SVD explicitly separates these two roles through the left and right singular vectors.

- **Left singular vectors (U)** capture outgoing connectivity patterns (hub-like behavior). Indeed, the leading right singular vector v_1 solves

$$v_1 = \arg\max_{\|v\|=1} \|Av\|,$$

which identifies the direction in node space that receives the largest total weighted influence. Nodes with large components in v_1 are therefore those that are strongly pointed to by important sources.

- **Right singular vectors (V)** capture incoming connectivity patterns (authority-like behavior). Indeed, the leading left singular vector u_1 satisfies

$$u_1 = \arg\max_{\|u\|=1} \|A^\top u\|,$$

highlighting nodes that strongly distribute influence to important destinations.

3.9 Low-Rank Approximations

Real-world networks are rarely random. Instead, they typically exhibit *mesoscopic structure* such as communities or repeated interaction patterns. These structures induce strong correlations among rows and columns of the adjacency matrix.

Instead of using only the top singular vector, one may retain the top k singular components:

$$A \approx U_k \Sigma_k V_k^\top.$$

Empirically, the singular values of many real networks decay rapidly, meaning that a small number of components captures most of the network's structural information.

A multi-dimensional importance score for node i can be defined as

$$\text{score}(i) = \sqrt{\sum_{j=1}^{k} \sigma_j^2 u_{ij}^2}.$$

Retaining only the top k singular values yields the best rank-k approximation to A in the Frobenius norm:

$$A_k = \arg\min_{\text{rank}(B)=k} \|A - B\|_F.$$

Rankings derived from the Singular Value Decomposition (SVD) identify nodes that contribute most strongly to the dominant connectivity patterns of a network. Large singular values correspond to important global structures, while nodes associated with large components in the corresponding singular vectors play a central role in shaping these structures.

Restricting the ranking to the leading singular components provides a natural way to reduce the impact of noise and to obtain more stable and robust

rankings. In empirical networks, edges may be noisy due to measurement errors, transient interactions, or missing data. Small singular values are highly sensitive to such perturbations, whereas the leading singular components are known to be stable under noise. As a result, focusing on the dominant singular vectors emphasizes persistent, large-scale connectivity patterns rather than local fluctuations.

Beyond robustness, SVD-based ranking captures multiple structural patterns simultaneously. A truncated SVD embeds the nodes of the network into a low-dimensional latent space of dimension k. In this representation, the coordinates of node i are given by the i-th row of $U_k\Sigma_k^{1/2}$ (or, equivalently, $V_k\Sigma_k^{1/2}$, depending on the chosen convention). Nodes with large norms in this latent space contribute strongly to the dominant connectivity modes of the network. Ranking nodes according to the magnitude of these coordinates therefore reflects their participation in the principal latent structures.

From this perspective, low-rank SVD-based ranking can be interpreted as a form of *spectral regularization*, where only the most informative and stable components of the network are retained to assess node importance.

Summary:

- Ranking arises from a recursive definition: important nodes are referenced by important nodes.
- The adjacency matrix encodes the graph's connectivity, enabling eigenvector-based ranking.
- Power iteration efficiently computes rankings even for very large graphs.
- PageRank generalizes eigenvector centrality by stabilizing the random walk.
- The dominant eigenvector of a stochastic or modified adjacency matrix provides the ranking scores.
- SVD ranking assesses node importance by measuring how strongly nodes contribute to the dominant latent connectivity patterns of a network, generalizing eigenvector centrality to both undirected and directed graphs.
- SVD ranking yields robust, low-rank rankings that emphasize stable, global structures while reducing sensitivity to noise and local fluctuations.

3.10 Questions

1. Explain how the degree of a node can be used as a first ranking metric.

2. Extend the previous measure by weighting neighbors by their own degree. Derive the formula for this second-order degree centrality.

3. Show how eigenvector centrality solves the equation $c = \alpha Ac$.

4. Explain why eigenvector centrality gives high score to nodes connected to high-score nodes.

5. Compute the eigenvector centrality of a simple 3-node chain.

6. Compare eigenvector centrality with SVD-based ranking using the first singular vector.

7. Why does normalizing the ranking vector matter?

8. Consider a food web adjacency matrix. Explain what a large centrality value means ecologically.

9. What biases might arise when ranking species only by network position?

10. What does a large singular value of an adjacency matrix indicate about the structure of a network?

11. Why do left and right singular vectors coincide in undirected networks, and why do they represent different notions of importance in directed networks?

12. What is the effect of retaining only the top k singular values and vectors when ranking nodes, and why does this lead to more robust rankings?

4 Linear Algebra Perspective of Principal Component Analysis

Principal Component Analysis (PCA) is a fundamental dimensionality-reduction technique grounded in linear algebra. Its goal is to identify the most informative directions in the data, i.e. those along which the data varies the most, and to express the data in a new coordinate system aligned with these directions.

4.1 Intuitive Explanation

Consider a dataset consisting of m observations, each described by n features. We store the data in a matrix

$$X \in \mathbb{R}^{m \times n},$$

Each row corresponds to an observation and each column to a feature. Geometrically, each observation is a point in $\mathbb{R}^n$. We assume that the data have been *centered*, i.e. each column has zero mean.

PCA aims at determining the directions in the feature space along which the data vary the most. In this context, the first principal component is the direction along which the projection of the data has maximal variance. The second principal component is the direction of maximal remaining variance, subject to being orthogonal to the first, and so on.

Thus, using the PCA one can determine: i) orthogonal directions capturing decreasing amounts of variance, ii) a low-dimensional subspace that best approximates the data.

Projecting the data onto these directions yields a new representation in which: i) the coordinates are uncorrelated, ii) most of the information is concentrated in the first (few) components.

4.2 Covariance Matrices and Variance Maximization

The sample covariance matrix of the features is

$$C = \frac{1}{m} X^\top X \in \mathbb{R}^{n \times n}.$$

For a unit vector $v \in \mathbb{R}^n$, the variance of the data projected onto v is:

$$\mathrm{Var}(Xv) = v^\top C v.$$

Then, maximizing the variance $\lambda = \mathrm{Var}(Xv)$ under the constraint $\|v\| = 1$ leads to the eigenvalue problem:

$$Cv = \lambda v.$$

Consequently, the eigenvectors of $X^\top X$ are the **principal directions** in feature space, and eigenvalues measure the amount of variance explained along each direction.

4.3 Formal PCA via Spectral Decomposition

Since $C = X^\top X$ is symmetric and positive semidefinite, the **spectral theorem** applies:

$$C = V \Lambda V^\top,$$

where:

- $V = [v_1, \ldots, v_n]$ is an orthonormal basis of eigenvectors,
- $\Lambda = \mathrm{diag}(\lambda_1 \geq \lambda_2 \geq \cdots \geq 0)$ contains the eigenvalues.

The k leading eigenvectors $V_k = [v_1, \ldots, v_k]$ define a k-dimensional subspace capturing the maximum variance, and the projection of the data onto this subspace is simply:

$$Z = XV_k,$$

Each row of Z represents an observation expressed in the new principal-component coordinates.

4.4 PCA and the Dual Formulation: $XX^\top$

When the number of features is very large ($n \gg m$), it is often more efficient to consider:

$$XX^\top \in \mathbb{R}^{m \times m}.$$

The matrices $X^\top X$ and $XX^\top$ share the same nonzero eigenvalues. If

$$X^\top X v = \lambda v,$$

then

$$u = \frac{1}{\sqrt{\lambda}} X v$$

satisfies

$$XX^\top u = \lambda u.$$

Interpretation:

- eigenvectors of $X^\top X$: principal directions in *feature space*,
- eigenvectors of $XX^\top$: principal directions in *observation space*.

This dual view is central in kernel PCA and high-dimensional data analysis.

4.5 Connection with Singular Value Decomposition

The SVD of the centered data matrix is:

$$X = U \Sigma V^\top.$$

Then:

$$X^\top X = V \Sigma^2 V^\top, \qquad XX^\top = U \Sigma^2 U^\top.$$

Thus:

- right singular vectors V are the principal directions,
- singular values satisfy $\sigma_i^2 = \lambda_i$,
- left singular vectors U give the coordinates of observations in PCA space.

This shows that PCA is a direct consequence of the spectral theorem applied to symmetric covariance matrices.

4.6 Feature Mapping and Dimensionality Reduction

Using the principal components, each observation $x \in \mathbb{R}^n$ is mapped to:

$$z = V_k^\top x \in \mathbb{R}^k.$$

This mapping:

- preserves maximal variance,
- removes redundancy and correlations,
- provides a compact and interpretable representation.

Summary:
PCA is an eigenvalue problem whose solution provides optimal linear coordinates for data representation. To do so it aims at:

- building a covariance matrix from the data,
- applying the spectral theorem,
- using eigenvectors as a new orthonormal basis,
- projecting the data onto the leading eigen-directions.

4.7 Questions

1. Why is it necessary to center the data matrix X before applying PCA? What would the principal components represent if the data were not centered?
2. Explain geometrically what a principal component represents in $\mathbb{R}^n$. How is it related to variance maximization?
3. Show that the variance of the projected data onto a unit vector v can be written as $v^\top C v$. Why does maximizing this quantity lead to an eigenvalue problem?
4. What is the interpretation of the eigenvalues of the covariance matrix $C = X^\top X$? How are they used to decide how many principal components to keep?

5. Why are the eigenvectors of C orthogonal? Which theorem guarantees this property?

6. Describe the difference between computing PCA using $X^\top X$ versus $XX^\top$. In which practical situation is the dual formulation preferable?

7. Given an eigenpair (λ, v) of $X^\top X$, explain how to construct the corresponding eigenvector of $XX^\top$. What is the geometric meaning of this transformation?

8. Explain how the Singular Value Decomposition of X directly yields the PCA solution. Which matrices in the SVD correspond to principal directions and variances?

9. Given a new observation $x \in \mathbb{R}^n$, explain how it is projected into a k-dimensional PCA space. What information is preserved and what is lost in this projection?

10. PCA produces uncorrelated features in the transformed space. Explain why this property follows from the diagonalization of the covariance matrix.

5 Computing Eigenvalues and Eigenvectors: Numerical Methods (Bonus)

Given a square matrix $A \in \mathbb{R}^{n\times n}$, finding its eigenpairs (λ, x) such that $Ax = \lambda x$ is central task for many applications. Exact symbolic solutions based on the computation of the characteristic polynomial roots are impractical for large n. Instead, one often relies on numerical algorithms.

5.1 The Characteristic Polynomial

Given a square matrix $A \in \mathbb{R}^{n\times n}$, the **characteristic polynomial** of A is defined as

$$p_A(\lambda) \;=\; \det(A - \lambda I).$$

The eigenvalues of A are exactly the real (or complex) roots of this polynomial. In principle, solving the eigenvalue problem is equivalent to computing $p_A(\lambda)$ and then finding its roots:

$$p_A(\lambda) = 0 \quad \Longleftrightarrow \quad \lambda \text{ is an eigenvalue of } A.$$

For very small matrices (e.g., 2×2 or 3×3), this approach is viable, and explicit formulas exist. In such settings, the characteristic polynomial gives direct, exact eigenvalues. Nevertheless, computing eigenvalues by forming and solving the characteristic polynomial is numerically unreliable and inefficient. The main reasons are:

1. **No stable general root-finding formulas for $n \geq 5$.** By the Abel–Ruffini theorem, there are no general closed forms, and numerical root-finding for high-degree polynomials is notoriously difficult.

2. **Catastrophic sensitivity to perturbations.** Small changes in the coefficients of a polynomial (due to rounding errors) can produce large changes in the roots, which makes the method unstable.

3. **Bad conditioning of the coefficient representation.** The coefficients of $p_A(\lambda)$ can be very large or very small, even when the entries of A are well-behaved.

4. **Computational cost.** Even forming the characteristic polynomial takes $O(n^4)$ operations in general, far more expensive than modern eigenvalue algorithms.

The characteristic-polynomial approach is a poor method for computing eigenvalues in practice. Modern numerical linear algebra instead uses iterative, stable, and structure-preserving algorithms, which offer both efficiency and numerical robustness.

5.2 The Rayleigh Quotient

The Rayleigh quotient of a nonzero vector $x \in \mathbb{R}^n$ with respect to a square matrix A is defined as

$$R_A(x) \;=\; \frac{x^\top A x}{x^\top x}.$$

It provides a scalar measure of how the matrix A stretches the direction defined by x. If A is symmetric, the Rayleigh quotient has interesting properties. Most importantly, for any eigenpair (λ, v) with $\|v\| = 1$, we have

$$R_A(v) = \lambda,$$

and for any nonzero x,

$$\lambda_{\min} \;\le\; R_A(x) \;\le\; \lambda_{\max}.$$

Thus the Rayleigh quotient always lies between the smallest and largest eigenvalues, and achieves these extrema *only* at the corresponding eigenvectors. This makes the Rayleigh quotient a fundamental tool in eigenvalue computation. Many numerical methods seek to iteratively improve a vector x_k so that $R_A(x_k)$ converges to the desired eigenvalue.

5.3 Power Iteration

The power iteration method (also known as the von Mises iteration) is the simplest numerical algorithm for approximating the *dominant* eigenvalue of a matrix—that is, the eigenvalue λ_1 of largest magnitude—and a corresponding eigenvector. It is particularly well suited for large, sparse, or structured matrices for which more sophisticated algorithms may be too costly.

Let $A \in \mathbb{R}^{n\times n}$ be a diagonalizable matrix whose eigenvalues satisfy

$$|\lambda_1| > |\lambda_2| \ge \cdots \ge |\lambda_n|.$$

The power iteration proceeds as follows:

1. **Initialization:** choose a random vector $b^{(0)}$ such that
$$\|b^{(0)}\|_2 = 1.$$
2. **Iterate:**
$$w^{(k)} = A b^{(k)}, \qquad b^{(k+1)} = \frac{w^{(k)}}{\|w^{(k)}\|_2}.$$
3. **Rayleigh quotient estimate of the eigenvalue:**
$$\lambda^{(k)} = (b^{(k)})^\top A b^{(k)}.$$

For sufficiently large k, the vector $b^{(k)}$ converges (up to normalization) to the dominant eigenvector v_1 if two conditions are fulfilled: i) the initial vector $b^{(0)}$ is not orthogonal to v_1. ii) the dominant eigenvalue λ_1 is unique in magnitude.

Intuition: Let us express the initial vector as a linear combination of eigenvectors:

$$b^{(0)} = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n, \qquad c_1 \neq 0.$$

Applying A repeatedly yields

$$A^k b^{(0)} = c_1 \lambda_1^k v_1 + c_2 \lambda_2^k v_2 + \cdots + c_n \lambda_n^k v_n.$$

Factorizing by λ_1^k gives

$$A^k b^{(0)} = \lambda_1^k \left(c_1 v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1} \right)^k v_2 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1} \right)^k v_n \right).$$

Since

$$\left| \frac{\lambda_i}{\lambda_1} \right| < 1 \quad \text{for all } i \geq 2,$$

the terms involving $v_2, \ldots, v_n$ decay geometrically as $k \to \infty$. Thus,

$$A^k b^{(0)} \longrightarrow \lambda_1^k \, c_1 \, v_1,$$

and after normalization the iteration converges to v_1.

Moreover, at each step, the algorithm computes the *Rayleigh quotient*:

$$\lambda^{(k)} = (b^{(k)})^\top A b^{(k)},$$

When $b^{(k)}$ approaches the true eigenvector v_1, this quotient converges to

$$\lambda_1 = v_1^\top A v_1.$$

The Rayleigh quotient thus provides a continuously improving eigenvalue estimate without solving any additional equations.

Advantages and Limitations: This method is simple to implement, very cheap per iteration (only one dot product), and it works well for large or sparse matrices. Nevertheless, it only computes the *dominant* eigenpair. Moreover it can become slow when the spectral gap $|\lambda_2/\lambda_1|$ is small. And it fails if the initial vector is orthogonal to v_1 (rare with random vectors) or if $|\lambda_1| = |\lambda_2|$ (no unique dominant eigenvalue).

5.4 Inverse Iteration and the Shifted Power Method

While the classical power iteration computes the *dominant* eigenvalue of a matrix, it does not provide access to other eigenvalues. To target a specific eigenvalue—for example the smallest one, or any eigenvalue near a given number—we use the **inverse iteration** or **shifted power method**. These methods rely on the idea that applying the power iteration to a suitably modified matrix can make any chosen eigenvalue behave like the "dominant" one.

Spectral Transformation by Shifting Let $A \in \mathbb{R}^{n\times n}$ be diagonalizable with eigen-decomposition

$$A = XDX^{-1}, \qquad D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

For a real number μ such that $\mu \neq \lambda_i$ for all i, consider the *shifted matrix*

$$A - \mu I.$$

Applying the shift inside the eigendecomposition yields

$$A - \mu I = XDX^{-1} - \mu I = XDX^{-1} - \mu XX^{-1} = X(D - \mu I)X^{-1}.$$

Taking the inverse, we obtain

$$(A - \mu I)^{-1} = X(D - \mu I)^{-1}X^{-1}.$$

Thus:

- A and $(A - \mu I)$ *share the same eigenvectors*;
- the eigenvalues of $(A - \mu I)^{-1}$ are $(\lambda_i - \mu)^{-1}$.

Because the reciprocals magnify eigenvalues near μ, the eigenvalue closest to μ becomes the one with *largest magnitude* in $(A - \mu I)^{-1}$. This means that the usual power iteration applied to $(A - \mu I)^{-1}$ will converge to the eigenvector corresponding to the eigenvalue of A closest to the shift μ.

Inverse Iteration Algorithm Given a shift μ close to a desired eigenvalue, the inverse iteration proceeds as:

1. **Choose** an initial vector $b^{(0)}$ with $\|b^{(0)}\|_2 = 1$.
2. **Iterate:**
$$w^{(k)} = (A - \mu I)^{-1}b^{(k)}, \qquad b^{(k+1)} = \frac{w^{(k)}}{\|w^{(k)}\|_2}.$$
3. **Estimate the eigenvalue** using the Rayleigh quotient:
$$\lambda^{(k)} = (b^{(k)})^\top A b^{(k)}.$$

When μ is a good approximation of an eigenvalue, the method converges rapidly.

Interpretation Inverse iteration is equivalent to applying classical power iteration to $(A - \mu I)^{-1}$:

$$b^{(k+1)} = \frac{(A - \mu I)^{-1}b^{(k)}}{\|(A - \mu I)^{-1}b^{(k)}\|}.$$

Because the eigenvalue $(\lambda_i - \mu)^{-1}$ is largest in magnitude when λ_i is closest to μ, the iterate $b^{(k)}$ converges to the corresponding eigenvector.

Advantages and Limitations This method is a simple extension of the power iteration framework that allows targeting *any* eigenvalue near a chosen shift μ. It tends to converge faster when μ is close to the true eigenvalue. Nevertheless, it requires solving a linear system $w^{(k)} = (A - \mu I)^{-1} b^{(k)}$ at each iteration, which may be costly. Moreover the method fails if μ exactly equals an eigenvalue (matrix becomes singular) and its convergence depends heavily on the quality of the shift.

Rayleigh Quotient Iteration The Rayleigh quotient iteration (RQI) is a refined version of inverse iteration in which the shift μ is not fixed but updated at every step using the *Rayleigh quotient* of the current iterate. Starting from a normalized random vector $v^{(0)}$, one computes an eigenvalue estimate

$$\lambda^{(0)} = (v^{(0)})^\top A v^{(0)},$$

then performs inverse iteration with the dynamically updated shift $\lambda^{(k)}$. At each iteration, the method solves

$$w^{(k)} = (A - \lambda^{(k)} I)^{-1} v^{(k)}, \qquad v^{(k+1)} = \frac{w^{(k)}}{\|w^{(k)}\|_2}, \qquad \lambda^{(k+1)} = (v^{(k+1)})^\top A v^{(k+1)}.$$

This adaptive shifting accelerates convergence, and turns it into one of the fastest algorithms for computing individual eigenvalues with high precision. Its main drawback is the need to solve a linear system at each step.

5.5 QR Algorithm and Schur Decomposition

A fundamental approach for computing *all* eigenvalues of a square matrix is based on the **QR algorithm** (Gram-Schmidt), whose theoretical foundation is the **Schur decomposition**.

Schur Decomposition

For any square matrix $A \in \mathbb{C}^{n \times n}$, there exists a unitary matrix Q and an upper triangular matrix T such that

$$A = QTQ^*.$$

Q is unitary: $Q^*Q = I$, T is upper triangular. And A and T are *similar* matrices, indeed, two square matrices $A, B \in \mathbb{C}^{n \times n}$ are said to be *similar* if there exists an invertible matrix $P \in \mathbb{C}^{n \times n}$ such that $B = P^{-1}AP$. The matrix P is called a *similarity transformation*. Similarity is an equivalence relation on square matrices and expresses the idea that A and B represent the same linear operator in two different bases. Similar matrices have the same eigenvalues (with the same algebraic multiplicities), the same determinant, trace, rank, and characteristic polynomial, therefore, A and T share the same eigenvalues.

Moreover, the eigenvalues of a triangular matrix are precisely its diagonal entries. Indeed, let $T \in \mathbb{C}^{n\times n}$ be an upper (or lower) triangular matrix:

$$T = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ 0 & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & t_{nn} \end{pmatrix}.$$

Then the eigenvalues of T are the diagonal entries $t_{11}, t_{22}, \ldots, t_{nn}$, each counted with its algebraic multiplicity. The characteristic polynomial of T is

$$\chi_T(\lambda) = \det(T - \lambda I).$$

Since $T - \lambda I$ is also triangular, its determinant is the product of its diagonal entries:

$$\det(T - \lambda I) = (t_{11} - \lambda)(t_{22} - \lambda)\cdots(t_{nn} - \lambda).$$

The roots of this polynomial are therefore the diagonal elements t_{ii}. These roots are exactly the eigenvalues of T.

A triangular matrix corresponds to a system in which each variable only depends on itself and on the previous ones. As a result, the action of the matrix on the canonical basis is almost diagonal, and the "scaling factors"—the eigenvalues—are directly visible on the diagonal.

Consequently, the Schur factorization reveals all eigenvalues of A, which appear on the diagonal of T.

QR Iteration

The QR algorithm provides an efficient iterative method to compute this Schur form. Starting from $A_0 = A$, each iteration consists of two steps:

$$A_k = Q_k R_k, \qquad \text{(QR factorization - Gram-Schmidt)}$$

$$A_{k+1} = R_k Q_k = Q_k^* A_k Q_k, \qquad \text{(similarity transformation)}.$$

By construction, each A_{k+1} is similar to A_k, and hence to A. As the iterations proceed, the matrices A_k converge toward an upper triangular matrix—the Schur form—whose diagonal entries approximate the eigenvalues of A. Moreover, the cumulative product $Q_1 Q_2 \cdots Q_k$ converges toward a unitary matrix whose columns approximate the eigenvectors.

In practice The QR Iteration computes *all* eigenvalues and eigenvectors of a dense matrix. It is numerically stable and robust in practice, and well suited for moderate-sized matrices. However its computational complexity is large and hence this method is not suitable for very large matrices. Moreover, it requires dense matrix operations; does not exploit sparsity unless optimized variants are used. Several improvements (e.g., Hessenberg reduction, Wilkinson shift, deflation method) accelerate the convergence and reduce the cost of the QR iteration method, turning it into the standard eigenvalue method in high-quality numerical linear algebra libraries (e.g., LAPACK).

5.6 Conclusion: Choosing the Right Eigenvalue Method

The computation of eigenvalues and eigenvectors is a central task in numerical linear algebra, and the choice of an appropriate method strongly depends on the structure, size, and properties of the matrix being analyzed. No single algorithm is universally optimal; instead, one must select a technique suited to the problem's scale and precision requirements.

Dominant Eigenpair Only. When the goal is to compute *only the dominant eigenvalue* (the one of largest magnitude) and its associated eigenvector, simple iterative methods such as the **Power Iteration** or the **Rayleigh Quotient Iteration** are often sufficient. These methods are easy to implement, require only matrix–vector products, and are particularly efficient for large sparse matrices. The Rayleigh quotient iteration, in particular, offers extremely fast convergence once close to an eigenvector.

A Few Eigenpairs. When only a small number of eigenvalues and eigenvectors are needed (for example, those closest to a given value), **Inverse Iteration** and the **Shifted Power Method** become more appropriate. By shifting the matrix and repeatedly solving linear systems, these methods target eigenpairs near a chosen shift, making them well-suited for problems such as computing the smallest eigenvalue or an interior eigenvalue.

All Eigenpairs of a Dense Matrix. For tasks requiring the full spectrum and all associated eigenvectors—typically in moderate-size dense matrices—the **QR Algorithm** (with shifts, Hessenberg reduction, and various optimizations) is the method of choice. It is robust, well-studied, and forms the foundation of many high-level linear algebra libraries.

Very Large or Sparse Matrices. When dealing with very large matrices, especially sparse ones, classical methods become too computationally expensive. Instead, **Krylov subspace methods**, such as the **Arnoldi iteration** (general case) and the **Lanczos algorithm** (symmetric case), are preferred. These methods construct a low-dimensional subspace that captures the action of the matrix, making it possible to approximate a few dominant eigenpairs efficiently without forming dense matrices.

Symmetric or Hermitian Matrices. Symmetric matrices enjoy many specialized, stable, and highly efficient algorithms. Methods such as **Lanczos**, the **symmetric QR iteration**, and **tridiagonal reduction** exploit symmetry to reduce computation and improve numerical stability.

Ill-Conditioned or Difficult Spectra. When the matrix is ill-conditioned, or when its eigenvalues are close to each other (i.e., small spectral gap), con-

vergence may become slow even for advanced methods. In such cases, two techniques can greatly improve performance:

- **Preconditioning**: Before applying an iterative method, the system is transformed into an equivalent form whose eigenvalues are more favorably distributed, leading to faster convergence.
- **Deflation**: Once an eigenpair is computed, the matrix is modified to remove its influence, allowing the algorithm to focus on the remaining spectrum without being affected by already-identified eigenvalues.

Eigenvalue Decomposition with Numerical Libraries Scientific computing libraries such as `NumPy`, `SciPy` do not implement eigenvalue algorithms directly in `Python` or other high-level programming languages (e.g., `MATLAB`, and `R`). Instead, they rely on highly optimized, compiled numerical libraries—primarily **LAPACK** (Linear Algebra PACKage) and its low-level BLAS routines.

For a general (non-symmetric) matrix, NumPy's function `np.linalg.eig` proceeds roughly as follows:

1. The matrix is first reduced to **upper Hessenberg form** (i.e., an "almost" triangular matrix that has zero entries below the first subdiagonal) using orthogonal transformations. This drastically reduces the computational cost while preserving eigenvalues.
2. The **QR algorithm with shifts** is then applied to the Hessenberg matrix.
3. The algorithm iterates until the matrix converges to **Schur form**, from which the eigenvalues (the diagonal entries) are extracted. If eigenvectors are requested, they are recovered by back-substitution.

For **symmetric or Hermitian** matrices, NumPy uses `np.linalg.eigh`, which takes advantage of symmetry:

1. The matrix is reduced to a **tridiagonal form** using Householder reflections. This drastically reduces computational requierements.
2. A specialized **symmetric QR algorithm** or other tools are applied to the tridiagonal matrix. These methods are faster and more stable than the general QR algorithm.

Summary. Eigenvalue computation relies on a broad family of algorithms, each suited to different matrix types and computational goals. Choosing the right method depends on:

- Matrix size
- Matrix sparsity
- Number of eigenpairs needed.
- Structural properties (symmetric, Hermitian)
- Numerical behavior (spectral gap, conditioning),

Scientic libraries rely on routines written in highly optimized Fortran or C, parallelized, and often use CPU-optimized BLAS instructions and rely on decades of numerical analysis research and highly engineered software

Summary Table

Method	**Computes**	**Best For**
Power iteration	Dominant eigenpair	large sparse, dominant eigenvalue
Inverse / shift iteration	Eigenpair near μ	targeted eigenpair
Krylov (Arnoldi / Lanczos)	few eigenpairs	very large sparse matrices
QR algorithm	all eigenpairs	small/medium dense matrices

5.7 Questions

1. Explain the idea of the power iteration and its main limitation.
2. Compute two iterations of the power method on $A = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$, starting from $b = (1, 1)^T$.
3. What is the Rayleigh quotient? Why does it improve eigenvalue estimates?
4. What is the goal of the QR algorithm?
5. Explain the Schur form and why it reveals the eigenvalues.
6. Which method is preferable for very large sparse matrices? Why?
7. Explain what *deflation* is in eigenvalue computation.
8. Explain what *preconditioning* is and why it may accelerate convergence.
9. Why do libraries reduce matrices to Hessenberg or tridiagonal form before QR iteration?
10. What risks exist when blindly trusting numerical eigenvalue outputs for sensitive decisions?

6 Practical Session Question

In [1], the authors investigate how to identify species that are critical for maintaining ecosystem functioning, with a particular focus on ecological food webs. Their approach is based on the idea that species importance can be quantified using network centrality measures, under the assumption that species occupying central positions in a food web exert a stronger influence on the rest of the ecosystem.

However, centrality alone does not fully capture ecological importance. A species may also be crucial because it occupies a unique position in the network—one that cannot be easily replaced if the species is lost. To address this, the authors hypothesize that ecosystem robustness requires an interplay between centrality and functional redundancy: species with high centrality should ideally have network positions that are shared by other species, so that their loss does not cause disproportionate disruption.

To explore these ideas, different food web datasets can be downloaded from the Web of Life ecological networks database (`https://www.web-of-life.es/`), which provides food webs from a wide range of ecosystems and geographical regions. These food webs are represented as adjacency matrices, where the entry at row i and column j denotes the relative frequency of prey species s_i in the diet of predator species s_j. Note that these matrices are generally weighted and not necessarily symmetric.

Using the linear algebra concepts introduced in this chapter (such as eigenvalue-based centrality measures and singular value decomposition) as well as concepts from statistics (such as correlation analysis) propose a methodology to quantify and compare the robustness of food webs across different ecosystems worldwide, justify it and apply it.

References

[1] S.-M. Lai, W.-C. Liu, and F. Jordán. On the centrality and uniqueness of species from the network perspective. *Biology Letters*, 8(4):570–573, 2012.