SIMCC 2026

A Linear Algebra Guide to the Strange Life of High-Dimensional Data

Preamble

Many of the most interesting questions in modern science and technology involve data that lives in enormous, high-dimensional spaces. A single photograph is a list of millions of pixel values. A medical sample can be summarised by thousands of measurements. The state of a growing population, a financial market, or a planet's climate at every observed location is described by long vectors of numbers. Even recommender systems and language models reason about you and the world live inside spaces with millions of dimensions. None of this can be drawn or visualised the way a familiar 2D plot can, but it can still be studied. The language we use to study it is linear algebra.

This competition is a guided introduction to one of linear algebra's most useful ideas: that even when a system or a dataset has thousands of dimensions, almost everything that matters about it is concentrated along a small number of special directions. Finding and reasoning about those directions is what underlies many of the modern tools you have already encountered, from image compression to search engines to genomics. Across the three questions that follow, you will discover this principle for yourself, prove the mathematics that makes it work, and apply it to a real scientific dataset. No prior linear algebra is assumed; the background sections will introduce everything you need as you go.

Each of the three questions shows this principle at work in a different setting. The first looks at a simple population-dynamics problem in which the long-term behaviour of the system can be read off, almost effortlessly, from a single special direction. It is a striking demonstration that a few directions can capture an entire system's fate. The second develops a general algorithmic recipe for finding these special directions in very large datasets, even when the dataset is not square. This is the engine behind how modern data analysis decomposes a complicated table of measurements into its few most important feature dimensions, without having to inspect every entry one by one. The third turns to a problem familiar in physics, chemistry, and biology, namely the analysis of a real spectroscopic dataset, and shows how the very same linear-algebraic ideas reveal the hidden structure of the samples that would otherwise be impossible to see by eye in the raw, high-dimensional data.

Instructions to the participants

This challenge is assessed in three components, totalling **80 points**: a written report (58 pts), an oral presentation (12 pts), and a Q&A session with the judges (10 pts).

1. **Written report [58 points].** Provide written solutions, with supporting figures and brief code excerpts where appropriate, for each question. Aim for clarity over comprehensiveness: a clean derivation that lands the key insight in half a page is far better than a lengthy write-up that buries it. The report must:
    - be submitted as a single `.pdf`, written in English;

- **not exceed 10 pages**, including any annexes or appendices (no cover page is required);
- be typeset in 12-point Times New Roman with single line spacing;
- show all mathematical symbols and code clearly and legibly.
- The 10-page cap is firm, so budget your space accordingly. The data-analysis portions of Question 3 typically warrant the largest share. Use the report to present derivations, results, and interpretation; lengthy code listings belong in the accompanying notebook (below) and should be referenced from the report rather than reproduced in full.

Source code and outputs. Submit your source code and outputs as a single Jupyter notebook (`.ipynb`). The input files for the challenge (e.g. `mystery_matrix1.npy`) do not need to be re-uploaded.

Record of AI usage. If any AI tool was used in preparing your submission, include a record of the prompts and the responses they produced, in `.pdf` or `.doc` format.

2. **Presentation [12 points].** Give a 15–20 minute presentation based on your solution. If time does not permit covering everything, focus on the parts of your solution that involved non-obvious choices, and trust the audience to have read the preamble and the background sections of the problem.
3. **Q&A [10 points].** A panel of judges will orally assess parts of your report and presentation. Be ready to explain and defend any claim, derivation, or numerical result that appears in your submission within the time allotted by the judges.

1 Linear Algebra of Markov Matrices [14 points]

Background: Vectors and Matrices

A **vector** is a list of numbers arranged in a column. For example,

$$\vec{P} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$$

could represent the fractional populations of two species, where p_1 and p_2 are numbers between 0 and 1. We call $\vec{P}$ a vector because it has a direction (with p_1 and p_2 as components along different directions) and a magnitude $\|\vec{P}\|$ defined as $\sqrt{p_1^2 + p_2^2}$.

A **matrix** is a rectangular array of numbers. We can multiply a matrix $\mathbb{M}$ by a vector to obtain a new vector:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}.$$

For example, you can verify that $\begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$.

Background: Eigenvectors and Eigenvalues

A vector $\vec{\nu}$ is called an **eigenvector** of a matrix $\mathbb{M}$ if multiplying by $\mathbb{M}$ only scales it without changing its direction:

$$\mathbb{M}\,\vec{\nu} = \lambda\,\vec{\nu}.$$

The scaling factor λ (a scalar, possibly negative) is called the **eigenvalue** corresponding to $\vec{\nu}$.

Example. Check that $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ is an eigenvector of $\begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix}$ with eigenvalue 3.

How to find eigenvalues. The condition $\mathbb{M}\,\vec{\nu} = \lambda\,\vec{\nu}$ rearranges to $(\mathbb{M} - \lambda I)\vec{\nu} = \vec{0}$, where $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ is the identity matrix. This has a non-trivial solution (i.e., $\vec{\nu} \neq \vec{0}$) only when

$$\det(\mathbb{M} - \lambda I) = 0.$$

For a 2×2 matrix, $\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$.

Setup

Consider two types of organisms, type 1 and type 2, competing for resources. We track their **fractional populations** p_1 and p_2 (each between 0 and 1), where "fractional" means each is measured as a fraction of the total. We observe the population at discrete generations $t = 0, 1, 2, 3, \ldots$.

The populations evolve according to the rule

$$\begin{pmatrix} p_1(t+1) \\ p_2(t+1) \end{pmatrix} = \underbrace{\begin{pmatrix} 1-\epsilon & 0 \\ \epsilon & 1 \end{pmatrix}}_{\mathbb{M}} \begin{pmatrix} p_1(t) \\ p_2(t) \end{pmatrix}, \tag{1}$$

where ϵ (a real number between 0 and 1) is a fixed positive constant; think of it as a small fraction such as 0.1. Writing out the matrix multiplication,

$$p_1(t+1) = (1-\epsilon)\, p_1(t), \qquad p_2(t+1) = \epsilon\, p_1(t) + p_2(t).$$

At each generation, a fraction ϵ of type-1 organisms convert into type-2 organisms.

We call $\mathbb{M}$ a **Markov matrix**, and the rule "the next state depends only on the current state, not on history" is called the **Markov property**.

1a) Conservation [1 point].

Task: Show that $p_1(t) + p_2(t) =$ constant for any time t.

1b) Eigenspectra [2 points].

Task: Compute the eigenvectors $\vec{\nu}_1, \vec{\nu}_2$ and eigenvalues λ_1, λ_2 of the Markov matrix $\mathbb{M}$ in Eqn (1).

1c) Linear Combinations [1 point].

Assume that $p_1(t=0) = \frac{1}{2}$.

Task: Write $\vec{P}(t=0)$ as a linear combination of the eigenvectors of $\mathbb{M}$. A **linear combination** of vectors $\vec{\nu}_1$ and $\vec{\nu}_2$ is any expression of the form $c_1\vec{\nu}_1 + c_2\vec{\nu}_2$, where c_1, c_2 are constants.

1d) Eigenvectors make time evolution easy [2 points].

Assume that $\vec{P}(t=0) = c_1\vec{\nu}_1 + c_2\vec{\nu}_2$, where $\vec{\nu}_1$ and $\vec{\nu}_2$ are the eigenvectors of $\mathbb{M}$.

Task: Write an expression for $\vec{P}(t)$ for $t > 0$, $t \in \mathbb{Z}$, in terms of $c_1, c_2, \vec{\nu}_1, \vec{\nu}_2$ and the eigenvalues λ_1, λ_2 of $\mathbb{M}$.

1e) Long-time behavior [2 points].

Tasks:

(i) Show that as $t \to \infty$, the population vector $\vec{P}$ approaches the eigenvector with the larger eigenvalue [1 point].

(ii) Explain how the structure of $\mathbb{M}$ leads to this asymptotic population [1 point].

1f) Validation [2 points].

Task: Validate your results from parts (d) and (e) numerically (for example, in `python`), assuming that $p_1(t=0) = \frac{1}{2}$.

1g) Time-varying conversion [2 points].

So far ϵ was constant. Suppose now that this conversion rate decreases over time.

Task: Describe how the result of part (d) changes if $\epsilon(t) = \epsilon_0 \exp(-t/t_0)$, where ϵ_0 and t_0 are real, positive numbers with $\epsilon_0 > 0$ and $t_0 > 0$.

1h) Validate again [2 points].

Task: Validate your results in (g) numerically, in a fashion similar to part (f).

2 The Power Method and Singular Value Decomposition [16 points]

In Question 1 you found the eigenvectors of a 2×2 Markov matrix by solving the characteristic polynomial ($\det(\mathbb{M} - \lambda I) = 0$) that was introduced in the eigenvalue background of Question 1. This approach works perfectly well for small matrices. But many real applications involve matrices with thousands or millions of rows and columns: search engines ranking web pages, image and video compression, gene-expression analysis, recommender systems. Computing the full eigendecomposition of such matrices is often impossible, and usually unnecessary. In most applications, only the leading one or few eigenvectors carry the structure of interest. The **Power Method** is the simplest algorithm that exploits this. It works because of the behaviour you observed in Question 1e: when you iterate a matrix on a vector, the dominant eigenvector takes over.

In this question you will (i) state and prove the Power Method, (ii) understand why a clean **spectral gap** is essential for the algorithm to work and to be numerically stable, and (iii) build the **Singular Value Decomposition (SVD)** as a generalisation of eigendecomposition to non-square matrices. The two symmetric matrices that appear along the way will return in Question 3 as the *covariance matrix* and the *Gram matrix*.

2a) The Power Method, formally [2 points].

Let $\mathbb{X}$ be a symmetric $n \times n$ matrix with eigenvalues

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$$

and a corresponding orthonormal basis of eigenvectors $\vec{\nu}_1, \vec{\nu}_2, \ldots, \vec{\nu}_n$ (so $\vec{\nu}_i \cdot \vec{\nu}_j = 0$ for $i \neq j$ and $\vec{\nu}_i \cdot \vec{\nu}_i = 1$). The **Power Method** starts from an arbitrary **unit vector** $\vec{x}_0$, that is, a vector of length one ($\|\vec{x}_0\| := \sqrt{\vec{x}_0 \cdot \vec{x}_0} = 1$), and iterates

$$\vec{x}_{k+1} = \frac{\mathbb{X}\,\vec{x}_k}{\|\mathbb{X}\,\vec{x}_k\|}, \qquad k = 0, 1, 2, \ldots$$

Task: Prove that $\vec{x}_k \to \pm\,\vec{\nu}_1$ as $k \to \infty$, provided $\vec{x}_0$ has a nonzero component along $\vec{\nu}_1$.

2b) Convergence rate and the spectral gap [2 points].

The proof in 2a tells you the Power Method converges, but how fast? And what determines its sensitivity to noise?

Task: Show that with each iteration, the orthogonal residual $\|\vec{r}_k\|$ shrinks, relative to the parallel component $|\vec{x}_k \cdot \vec{\nu}_1|$, by at least a factor of $|\lambda_2/\lambda_1|$.

> We call $|\lambda_1| - |\lambda_2|$ (or equivalently the ratio $|\lambda_1/\lambda_2|$) the **spectral gap** of $\mathbb{X}$. A large spectral gap means the Power Method converges quickly and is robust to small perturbations. A small spectral gap means convergence is sluggish and the algorithm is vulnerable to floating-point noise: every iteration introduces rounding errors of order 10^{-16} in standard double-precision arithmetic, and these errors compete with the signal $\propto |\lambda_2/\lambda_1|^k$.

For a $10^6 \times 10^6$ matrix, computing the full eigendecomposition is computationally infeasible. But if the spectral gap is large, the Power Method finds $\vec{\nu}_1$ in just a few dozen matrix-vector multiplications, which is cheap even for huge sparse matrices. This is why the Power Method (and its descendants such as Lanczos and Arnoldi iteration) underlies how Google's PageRank, large-scale recommender systems, and modern PCA are computed in practice.

2c) When the spectral gap closes [3 points].

What happens when there is no spectral gap, i.e. $|\lambda_1| = |\lambda_2|$?

Tasks:

(i) Consider the explicit 2×2 matrix

$$\mathbb{X} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

which has eigenvalues $\lambda_1 = 1$ and $\lambda_2 = -1$, so $|\lambda_1| = |\lambda_2|$. Starting from a generic initial unit vector $\vec{x}_0 = (a, b)$ with both $a, b \neq 0$, compute $\vec{x}_k$ exactly for arbitrary k and show that the iteration does not converge to a single vector [1 point].

(ii) Explain in your own words why a vanishing spectral gap is a fundamental obstruction to the Power Method, not merely a slow-convergence problem [1 point].

(iii) Discuss briefly: in finite-precision arithmetic, what additional risks arise when $|\lambda_2/\lambda_1|$ is close to (but not exactly) 1 [1 point]?

2d) Two symmetric matrices: $\mathbb{X}^\top\mathbb{X}$ and $\mathbb{X}\mathbb{X}^\top$. [5 points]

The Power Method requires a symmetric matrix, but most data matrices in practice are rectangular: an $N \times M$ matrix with $N \neq M$ has no eigenvalues at all, because eigenvectors require the matrix to map a vector back to a vector of the same size. To address this, we build symmetric matrices out of the rectangular one.

Background: transpose and matrix multiplication. The **transpose** of a matrix A, written $A^\top$, is obtained by swapping rows with columns: $(A^\top)_{ij} = A_{ji}$. So a $p \times q$ matrix becomes a $q \times p$ matrix when transposed. **Matrix multiplication** extends the matrix-vector product you saw in Question 1's background. For matrices A of size $p \times q$ and B of size $q \times r$, the product AB is the $p \times r$ matrix whose (i, k) entry is the dot product of row i of A with column k of B:

$$(AB)_{ik} \;=\; \sum_{j=1}^{q} A_{ij} B_{jk}.$$

The number of columns in A must match the number of rows in B for the product to be defined.

Given any $N \times M$ matrix $\mathbb{X}$, consider the two products

$$\mathbb{X}^\top\mathbb{X} \quad (M \times M) \qquad \text{and} \qquad \mathbb{X}\mathbb{X}^\top \quad (N \times N).$$

(In the first product, $\mathbb{X}^\top$ is $M \times N$ and $\mathbb{X}$ is $N \times M$, so the result is $M \times M$; in the second, $\mathbb{X}$ is $N \times M$ and $\mathbb{X}^\top$ is $M \times N$, so the result is $N \times N$.)

Background: symmetric and positive semi-definite matrices.
A square matrix A is **symmetric** if it equals its own transpose, $A = A^\top$, or in entries $A_{ij} = A_{ji}$ for every pair of indices i, j. The most useful property of a real symmetric matrix is the **spectral theorem**, which we will take as a given fact: every real symmetric matrix has only real eigenvalues, and admits a complete orthonormal basis of real eigenvectors. This is the setting that the Power Method in 2a assumed.
A real symmetric matrix A is called **positive semi-definite (PSD)** if, for every real vector $\vec{w}$,

$$\vec{w}^\top A \, \vec{w} \;\geq\; 0.$$

A PSD matrix has non-negative eigenvalues. The proof is short: if $A\vec{v} = \lambda \, \vec{v}$ with $\vec{v}$ a unit eigenvector, then

$$\vec{v}^\top A \, \vec{v} \;=\; \vec{v}^\top (\lambda \vec{v}) \;=\; \lambda \, \|\vec{v}\|^2 \;=\; \lambda,$$

and the PSD inequality forces $\lambda \geq 0$. So a real symmetric matrix that is also PSD has eigenvalues that are real (from symmetry) and non-negative (from PSD).
Below you will show that $\mathbb{X}^\top \mathbb{X}$ and $\mathbb{X}\mathbb{X}^\top$ are both symmetric and PSD, so their eigenvalues are real and ≥ 0. We can therefore take square roots safely, which is how singular values are defined. The same guarantee will apply to the covariance and Gram matrices in Question 3, which are scalar multiples of these two products.

Tasks:

(i) Show that both $\mathbb{X}^\top \mathbb{X}$ and $\mathbb{X}\mathbb{X}^\top$ are symmetric and positive semi-definite. Combined with the background note above, this guarantees that all their eigenvalues are real and non-negative [2 points].

(ii) Order the eigenvalues of $\mathbb{X}^\top \mathbb{X}$ as $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M \geq 0$ (they are real and non-negative by the previous task). Define the **singular values** of $\mathbb{X}$ by $\sigma_i := \sqrt{\lambda_i}$, so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$. Prove that $\mathbb{X}^\top \mathbb{X}$ and $\mathbb{X}\mathbb{X}^\top$ have the same nonzero eigenvalues, both equal to σ_i^2 [3 points].

The Singular Value Decomposition. The pieces you assembled in 2d combine into a remarkable result: *every* $N \times M$ matrix $\mathbb{X}$ admits a decomposition

$$\mathbb{X} \;=\; \sum_{i=1}^{r} \sigma_i \, \vec{u}_i \, \vec{v}_i^{\top} \qquad \text{or, equivalently,} \qquad \mathbb{X} \;=\; \mathbb{U}\,\Sigma\,\mathbb{V}^{\top},$$

where r is the number of nonzero singular values σ_i, the unit vectors $\vec{v}_i \in \mathbb{R}^M$ are eigenvectors of $\mathbb{X}^{\top}\mathbb{X}$ (the **right singular vectors**), and the corresponding $\vec{u}_i := \mathbb{X}\vec{v}_i/\sigma_i \in \mathbb{R}^N$ are eigenvectors of $\mathbb{X}\mathbb{X}^{\top}$ (the **left singular vectors**). The matrix Σ is diagonal with the singular values down its diagonal, and $\mathbb{U}$ and $\mathbb{V}$ collect the left and right singular vectors as their columns. This is the **Singular Value Decomposition** (SVD), and it generalises eigendecomposition to every matrix, square or rectangular.

Geometric picture. Because $\mathbb{U}$ and $\mathbb{V}$ are orthogonal (their columns are unit vectors that are mutually perpendicular), they act on space as rigid rotations or reflections, while Σ acts as a stretch or shrink along each axis independently. So multiplying any vector by $\mathbb{X}$ amounts to three geometrically simple steps in sequence: *rotate* the input space ($\mathbb{V}^{\top}$), *stretch* along each new axis by the corresponding factor σ_i (Σ), and *rotate* the result into the output space ($\mathbb{U}$). Pictorially, $\mathbb{X}$ sends the unit sphere in $\mathbb{R}^M$ to an ellipsoid in $\mathbb{R}^N$ whose semi-axis lengths are exactly the singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$. The right singular vectors $\vec{v}_i$ are the input directions that get stretched the most, and the left singular vectors $\vec{u}_i$ are where those directions end up after the rotation, scaling, and second rotation are applied.

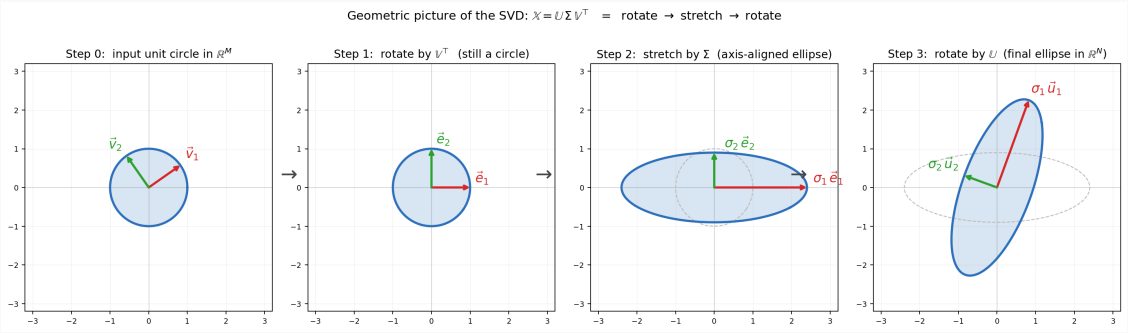

Figure: the rotate–stretch–rotate picture for a 2×2 example. The unit circle in the input space is first rotated by $\mathbb{V}^{\top}$ (carrying the right singular vectors $\vec{v}_i$ onto the coordinate axes), then stretched by Σ along each axis to produce an axis-aligned ellipse with semi-axes σ_1, σ_2, and finally rotated by $\mathbb{U}$ into the output space, where the ellipse's semi-axes become $\sigma_1\vec{u}_1$ and $\sigma_2\vec{u}_2$.

2e) Power Method for SVD: a complete worked example [4 points].

The Power Method gives the largest eigenvalue and corresponding eigenvector of any symmetric matrix. Combined with the construction in 2d, this gives a way to find the largest singular value σ_1 of any rectangular matrix $\mathbb{X}$ without computing the full eigendecomposition. Concretely, applying the Power Method to $\mathbb{X}^{\top}\mathbb{X}$ returns the largest eigenvalue $\lambda_1 = \sigma_1^2$ and a corresponding unit eigenvector $\vec{v}_1 \in \mathbb{R}^M$. The companion vector $\vec{u}_1 \in \mathbb{R}^N$ can then be obtained by a single matrix–vector product:

$$\vec{u}_1 \;:=\; \frac{\mathbb{X}\,\vec{v}_1}{\sigma_1}.$$

Tasks:

(i) Present a step-by-step algorithm, written entirely in clear language-agnostic **pseudocode**, that takes an arbitrary $N \times M$ matrix $\mathbb{X}$ as input and returns σ_1, $\vec{v}_1$, and $\vec{u}_1$ using only the Power Method and matrix-vector multiplications. State the inputs, outputs, loop structure, and termination condition explicitly, and use mathematical notation (such as $\mathbb{X}^\top \vec{y}$, $\|\vec{y}\|$) rather than language-specific syntax. Decide whether to apply the Power Method to $\mathbb{X}^\top \mathbb{X}$ or to $\mathbb{X}\mathbb{X}^\top$, and justify your choice in terms of computational cost when $N \ll M$ (or when $M \ll N$) [2 points].

(ii) Apply your algorithm *by hand* (no computer) to the rectangular matrix

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

Show your working, find σ_1, $\vec{v}_1$, and $\vec{u}_1$, and verify by direct calculation that $\mathbb{X}\,\vec{v}_1 = \sigma_1\,\vec{u}_1$ and $\mathbb{X}^\top\,\vec{u}_1 = \sigma_1\,\vec{v}_1$ [2 points].

What to submit. Part (i) must be written in **pseudocode** only: no Python, no other programming language. The pseudocode should be language-agnostic, with each line a clear instruction in plain mathematical notation (assignments, loops, conditionals, vector and matrix operations) rather than the syntax of any specific language. Part (ii) is a hand calculation, with all steps shown; you may use a computer afterwards to verify your final values, but the working itself must be hand-derived.

The two symmetric matrices in 2d will return in Question 3 under their proper names. When the rows of $\mathbb{X}$ represent measurements (samples) and the columns represent measured quantities (features), $\mathbb{X}^\top \mathbb{X}$ is called the **covariance matrix** (it lives in feature space and tells you which features co-vary), and $\mathbb{X}\mathbb{X}^\top$ is called the **Gram matrix** (it lives in sample space and tells you which measurements are similar to one another). Concretely, the eigenvalues λ_i of $\mathbb{X}^\top \mathbb{X}$ that you ordered in 2d become the eigenvalues of the covariance matrix in Question 3, up to the normalisation $1/\widetilde{N}$. The fact that $\mathbb{X}^\top \mathbb{X}$ and $\mathbb{X}\mathbb{X}^\top$ share the same nonzero eigenvalues, which you established in 2d, is the reason both views of the data carry the same essential information; choosing between them is a question of computational cost and numerical conditioning.

3 Design Matrix, Covariance Matrix, Gram Matrix

In science and machine learning, we often collect data in a table of N measurements (rows), each recording M different quantities (columns). For example, you might have N vials of molecules, and for each vial record its spectroscopic response at M different light frequencies. Each row is one vial; each column is one frequency.

We call this table the **design matrix** $\mathbb{X}$, with entries X_{ij}, where:

- $i \in \{1, 2, \ldots, N\}$ indexes the **samples** (rows),
- $j \in \{1, 2, \ldots, M\}$ indexes the **features** (columns),
- $N = 1000$ and $M = 3041$.

The dataset for this question is provided as the file `mystery_matrix1.npy`. Throughout Question 3, every reference to "the design matrix $\mathbb{X}$" refers to the contents of this file. Each row could represent a particular infrared spectroscopic measurement of a particular vial of molecules from a food taste test (N different vials), where the M different columns of that row are the spectroscopic response of the molecules at M distinct frequencies.

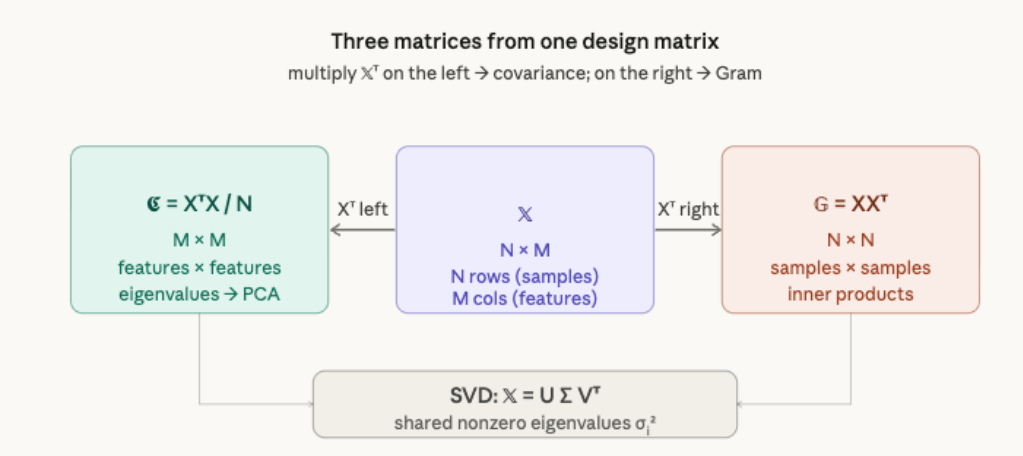

Figure: schematic of the design matrix $\mathbb{X}$ (file `mystery_matrix1.npy`). Each of the N rows is one sample's measurements across the M features; each of the M columns is one feature recorded across all N samples. The left box highlights one row $\vec{x}_i^{\top}$ (a single sample), and the right box highlights one column $\vec{x}^{(j)}$ (a single feature across all samples).

> Data collected in a lab is rarely clean. Before we can learn anything meaningful from $\mathbb{X}$, we must identify and remove measurement errors. In the parts below, you will clean the data, compress it using linear algebra, and uncover hidden structure within it. This is the workflow a practising data scientist would follow.

3a) Remove errors [2 points].

One type of error is that $\mathbb{X}$ contains T corrupted measurements where no sample molecules were in the vial, so the measurement records nothing but low background noise. Your first task is to find and remove all T of these measurements.

Tasks:

(i) How many measurements were corrupted? Explain your strategy for removing these measurement errors, and how you validated that they were removed [1 point].

(ii) Can you find the other error [1 point]?

3b) Compute variance [3 points].

> **Reminder: what is variance?** The variance of a feature j across all $\widetilde{N}$ measurements is how much that feature fluctuates from sample to sample. A feature that does not change between measurements has zero variance, telling you nothing about differences between samples.

Once the errors of the raw design matrix have been removed, we **mean-center** the data so that each feature has zero average across all measurements. For each column (feature) j, compute the mean across the N' remaining measurements,

$$\bar{X}_j = \frac{1}{N'} \sum_{i=1}^{N'} X_{ij},$$

and subtract $\bar{X}_j$ from every entry in column j. Call the resulting (error-removed, mean-centered) matrix $\widetilde{\mathbb{X}}$, with $\widetilde{N} = N'$ rows and M columns.

> **Why mean-center?** The covariance matrix below measures how features fluctuate around their average. Without this step, the leading eigenvector would be dominated by the average spectrum (which is shared by every sample) rather than by the directions in which samples genuinely differ. It is the differences between samples that we care about.

The **covariance matrix** of $\widetilde{\mathbb{X}}$ is

$$\mathbb{C} = \frac{1}{\widetilde{N}} \widetilde{\mathbb{X}}^\top \widetilde{\mathbb{X}}, \qquad C_{ij} = \frac{1}{\widetilde{N}} \sum_{k=1}^{\widetilde{N}} \widetilde{X}_{ki} \widetilde{X}_{kj}.$$

This is an $M \times M$ matrix. The diagonal entry C_{jj} is exactly the variance of feature j.

> **Sanity check.** Recall from Question 1 that the Markov matrix conserved $p_1 + p_2$. Something analogous holds here: the trace of $\mathbb{C}$ (the sum of diagonal entries) equals the total variance of your dataset, no matter how you later rotate or compress the data.

Tasks:

(i) Which single feature j^* has the largest variance? Report the index j^* and its variance value [1 point].

(ii) Plot the variance C_{jj} as a function of j [1 point].

(iii) Does the distribution of variance across features tell you anything interesting about the structure of the data [1 point]?

3c) Computing the Eigenspectrum [4 points].

Computing the eigenvectors and eigenvalues of $\mathbb{C}$ is equivalent to **Principal Component Analysis (PCA)**, one of the most widely used tools in data science. As in Question 1, where

the eigenvectors revealed the long-time behaviour of the Markov chain, the eigenvectors of $\mathbb{C}$ reveal the dominant patterns of variation in your data.

Eigenvalues are variances. For any unit vector $\vec{w} \in \mathbb{R}^M$, projecting each (mean-centered) measurement $\widetilde{\vec{x}}_i$ onto $\vec{w}$ produces a scalar $z_i = \widetilde{\vec{x}}_i \cdot \vec{w}$. The variance of these projected scalars across the $\widetilde{N}$ samples is

$$\mathrm{Var}(z) \;=\; \frac{1}{\widetilde{N}} \sum_{i=1}^{\widetilde{N}} z_i^2 \;=\; \vec{w}^{\top} \mathbb{C}\, \vec{w}.$$

So $\vec{w}^{\top}\mathbb{C}\,\vec{w}$ is the variance of the data when projected onto the linear combination of features specified by $\vec{w}$. Now choose $\vec{w} = \vec{\nu}_k$ to be a unit eigenvector of $\mathbb{C}$ with eigenvalue λ_k. Then

$$\vec{\nu}_k^{\top} \mathbb{C}\, \vec{\nu}_k \;=\; \vec{\nu}_k^{\top} (\lambda_k \vec{\nu}_k) \;=\; \lambda_k.$$

The eigenvalue λ_k is therefore the variance of the data when projected onto the k-th eigenvector. Diagonalising $\mathbb{C}$ (i.e. finding its eigenvectors) is the process of identifying orthogonal directions in feature space along which the data's total variance is decomposed, ranked from largest (λ_1) to smallest (λ_M). The leading eigenvector $\vec{\nu}_1$ identifies the direction of greatest variation, which you will examine in 3d.

The eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M \geq 0$ tell you how much variance is explained by each eigenvector direction. A large gap between λ_1 and λ_2 means one direction of variation dominates; the data has a strong principal axis.

Hint. Use `numpy.linalg.eigh` rather than `numpy.linalg.eig` for symmetric matrices: it is faster, numerically more stable, and guarantees real eigenvalues.

Sanity check. Verify that $\sum_{j=1}^{M} \lambda_j \approx \mathrm{tr}(\mathbb{C})$, confirming that the eigendecomposition is correct.

Tasks:

(i) Report the ratio λ_1/λ_2 [1 point].

(ii) Plot the eigenvalue spectrum (λ_k vs. k) on both linear and log scales [1 point].

(iii) The eigenvalue spectrum has an **elbow**: a small number of large "signal" eigenvalues followed by a long flat tail of small "noise" eigenvalues. Identify the elbow $k^{\star}$ on your log-scale plot. Justify your answer [1 point].

(iv) What does a large ratio tell you about the structure of the data [1 point]?

3d) Interpreting the Leading Eigenvector [4 points].

The eigenvector $\vec{\nu}_1$ corresponding to λ_1 is an M-dimensional vector. It lives in feature space, the same space as the columns of $\widetilde{\mathbb{X}}$.

Connecting to Question 1. In Question 1, the eigenvector with eigenvalue 1 was the long-time steady state of the Markov dynamics: the direction the system "wanted" to be in. Here, $\vec{\nu}_1$ is the direction in feature space along which your measurements vary the most. It is the linear combination of the M spectral frequencies that best discriminates between different samples.

A concrete way to think about it. If you had to summarise each measurement $\vec{x}_i$ (a row of $\widetilde{\mathbb{X}}$) by a single number, the most information-preserving choice is the projection $z_i = \vec{x}_i \cdot \vec{\nu}_1$. The eigenvector $\vec{\nu}_1$ tells you which weighted combination of frequencies that single number corresponds to.

Tasks:

(i) Plot $\vec{\nu}_1$ as a function of feature index j (i.e., plot the components of the leading eigenvector). Interpret what you see: which regions of the frequency axis are most heavily weighted [1 points]?

(ii) Does $\vec{\nu}_1$ resemble any of the raw measurements of $\widetilde{\mathbb{X}}$ [2 points]?

3e) Discovering Clusters via the Covariance Matrix [2 points].

Rather than summarising each measurement by a single number, we can project onto the top 4 eigenvectors to get a 4-dimensional compressed representation:

$$\vec{z}_i = \begin{pmatrix} \vec{x}_i \cdot \vec{\nu}_1 \\ \vec{x}_i \cdot \vec{\nu}_2 \\ \vec{x}_i \cdot \vec{\nu}_3 \\ \vec{x}_i \cdot \vec{\nu}_4 \end{pmatrix} \in \mathbb{R}^4.$$

This gives a compressed matrix $\mathbb{Z}$ of size $\widetilde{N} \times 4$. In matrix notation, $\mathbb{Z} = \widetilde{\mathbb{X}}\,\mathbb{V}_4$, where $\mathbb{V}_4$ is the $M \times 4$ matrix whose columns are $\vec{\nu}_1, \vec{\nu}_2, \vec{\nu}_3, \vec{\nu}_4$.

Plotting tip. There are $\binom{4}{2} = 6$ possible pairs of projections. Arrange them in a grid. Use small, semi-transparent points (`alpha=0.4`, `s=10`) so overlapping clusters remain visible.

Tasks:

(i) Project $\widetilde{\mathbb{X}}$ onto the top 4 eigenvectors of $\mathbb{C}$ and produce the 6 pairwise 2D scatter plots [1 point].

(ii) How many distinct clusters of measurements do you see? In this context, each cluster of measurements should correspond to a particular concoction of molecules [1 point].

3f) Discovering Clusters via the Gram Matrix [4 points].

The **Gram matrix** is defined as

$$\mathbb{G} = \frac{1}{\widetilde{N}}\,\widetilde{\mathbb{X}}\,\widetilde{\mathbb{X}}^{\top}, \qquad G_{ij} = \frac{1}{\widetilde{N}} \sum_{k=1}^{M} X_{ik}\, X_{jk}.$$

This is an $\widetilde{N} \times \widetilde{N}$ matrix. The entry G_{ij} measures the dot-product similarity between measurement i and measurement j. Each row of $\mathbb{G}$ is a similarity fingerprint of one measurement: how similar is this vial to every other vial?

Key contrast with $\mathbb{C}$. The covariance matrix $\mathbb{C}$ lives in feature space ($M \times M$) and tells you which features co-vary. The Gram matrix $\mathbb{G}$ lives in sample space ($\widetilde{N} \times \widetilde{N}$) and tells you which measurements are similar to one another. Both encode related information; in fact, as you will show in 3g, they share the same nonzero eigenvalues.

Connecting to Question 2. The covariance and Gram matrices share the same nonzero eigenvalues, exactly as you established in Question 2d. The eigenvectors of $\mathbb{G}$ live in sample space, while those of $\mathbb{C}$ live in feature space, so the two scatter pictures will be related by the same up-to-rescaling correspondence that links a unit eigenvector $\vec{v}_i$ of $\mathbb{C}$ to a unit eigenvector $\vec{u}_i := \widetilde{\mathbb{X}}\vec{v}_i/\sigma_i$ of $\mathbb{G}$.

Tasks:

(i) Compute the top 4 eigenvectors of $\mathbb{G}$ and project the measurements onto them. The eigenvectors of $\mathbb{G}$ are already $\widetilde{N}$-dimensional, so each component directly gives the coordinate of one measurement [2 points].

(ii) Produce the same set of 6 pairwise 2D scatter plots as in 3e [1 point].

(iii) How many clusters do you see [1 point]?

3g) Comparing the Two Approaches [3 points].

The mathematical connection. Suppose $\vec{\nu}$ is an eigenvector of $\mathbb{C} = \frac{1}{\widetilde{N}}\widetilde{\mathbb{X}}^\top\widetilde{\mathbb{X}}$ with eigenvalue λ. Then $\widetilde{\mathbb{X}}\vec{\nu}$ is an eigenvector of $\mathbb{G} = \frac{1}{\widetilde{N}}\widetilde{\mathbb{X}}\widetilde{\mathbb{X}}^\top$ with the same eigenvalue λ. (You can verify this by substituting directly.) Both matrices share the same nonzero eigenvalues: they are two views of the same underlying structure.

Yet the clusters may look different. Even if the eigenvalues match, the eigenvectors live in different spaces (M-dimensional feature space vs. $\widetilde{N}$-dimensional sample space), and numerical differences in how the two matrices are computed and diagonalised can affect the projections. There is also a subtler reason: one of the two matrices may have a more favourable condition number, making its eigenvectors easier to estimate accurately.

Tasks: Are the clusters you found in 3e and 3f the same? Specifically:

(i) Do you find the same number of clusters [1 point]? Can you explain this result [1 point]?

(ii) The *conceptual machinery* for clustering in 3e and 3f is different. In your own words, explain what it means for a measurement to belong to a cluster (a) when clustering is performed in the **covariance space** (top-k eigenvectors of $\mathbb{C}$, as in 3e), and (b) when clustering is performed in the **Gram space** (top-k eigenvectors of $\mathbb{G}$, as in 3f). What does each approach use as the "coordinates" of a measurement, and what does proximity in those coordinates mean about the underlying vials [1 point]?

3h) The forgetful taste tester [6 points].

These measurements came from 1000 vials of food samples in a very large blind taste test. The taste tester forgot the ground-truth labels for each sample.
The taste tester knows that there were fewer than 100 unique samples among the 1000 vials (each vial contains only one unique sample). The unique samples included fermented caffeinated drinks such as kombucha, cocoa drinks, coffee liqueurs, and Chinese tea.
The taste tester also suspects that there were ten dominant molecules in these samples: water, ethanol, methanol, caffeine, propionic acid, formic acid, lactic acid, glycerol, acetaldehyde, and acetic acid. The infrared spectroscopy was specifically designed to detect signatures of these molecules to fingerprint each unique sample.

Tasks:

(i) How many unique food samples were there actually? Justify your answer [4 points].

(ii) Which aspects of the previous parts of this competition were useful for arriving at the answer [2 points]?