# Programming Machine Learning Algorithms for HPC

- Linear Algebra

Dr. Pierrick Pochelu and Dr. Oscar J. Castro-Lopez

UNIVERSITÉ DU LUXEMBOURG

# Linear algebra

Introduction

# Why linear algebra is important?

- Linear algebra is one of the most fundamental and versatile subjects in mathematics.
- The practical skills learned by studying linear algebra—such as manipulating vectors and matrices—form an essential foundation for numerous applications in physics, computer science, statistics, **machine learning**, engineering, and many other fields of scientific and technological study.
- It underpins many modern advancements, from algorithms powering **artificial intelligence** to **data modeling** in **statistics**.

# Linearity

- It is the core of linear algebra.

A function f is linear if it obeys the following properties:

Additivity: $f(x_1 + x_2) = f(x_1) + f(x_2)$

Homogeneity (scalar multiplication): $f(a \cdot\ x_1) = a \cdot f(x_1)$

$$f(ax_1 + bx_2) = \mathrm{a}f(x_1) + bf(x_2)$$

- Linear functions transform a linear combination of inputs into the same linear combination of outputs.

# Why Linearity Matters

- **Simplicity**: Linear systems are much easier to solve, analyze, and understand than nonlinear systems. Linearity ensures that superpositions of solutions are also solutions, making problems more tractable.
- **Approximation**: Many complex, nonlinear systems can be approximated locally by linear systems. For instance, in calculus, nonlinear functions are approximated by their linear tangents through linearization.
- **Applications**: Linearity plays a key role in many fields. In physics, for example, linearity is central in quantum mechanics and wave theory. In machine learning, **linear models** like linear regression form the basis for more complex models.

# Definitions

- Vectors and matrices are the objects of study in linear algebra.
- A scalar is a number.
- A vector is a list of numbers.
- A matrix is a rectangular array of numbers with $m$ rows and $n$ columns.

Numbers can be naturals, integers, rationals, and real numbers.

# Vector operations $\vec{v}$

- Addition (denoted +)
- Subtraction, the inverse of addition (denoted -)
- Scaling (denoted implicitly)
- Dot product (denoted ·)
- Cross product (denoted $\times$)
- Length (denoted$||\vec{v}||$)

# Vector operations

Consider two vectors of three dimensions:

$$\vec{u} = (u_1, u_2, u_3) \text{ and } \vec{v} = (v_1, v_2, v_3)$$

$$\alpha \text{ is an arbitrary constant}$$

- $\vec{u} + \vec{v} = (u_1 + v_1, u_2 + v_2, u_3 + v_3)$
- $\vec{u} - \vec{v} = (u_1 - v_1, u_2 - v_2, u_3 - v_3)$
- $\alpha\vec{v} = (\alpha v_{1,} \alpha v_{2,} \alpha v_{3,})$
- $\vec{u} \cdot \vec{v} = (u_1 v_1 + \ u_2 v_2 + u_3 v_3)$
- $\vec{u} \times \vec{v} = (u_2 v_3 - u_3 v_2, u_3 v_2 - u_3 v_1, u_1 v_2 - u_2 v_1)$
- $||\vec{v}|| = \sqrt{v_1^2 + v_2^2 + v_1^3}$

# Matrix operations

- Addition (denoted A + B)
- Subtraction, the inverse of addition (denoted A - B)
- Scaling by a constant $\alpha$ (denoted $\alpha$A)
- Matrix product (denoted AB)
- Matrix-vector product (denoted $A\vec{v}$)
- Matrix inverse (denoted $A^{-1}$)
- Trace (denoted Tr(A))
- Determinant (denoted det(A) or |A|)

# Matrix addition and subtraction

The matrix addition and subtraction operations take pairs of matrices as inputs and produce matrices as outputs:

- $+: \mathbb{R}^{m \times n} \times \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$
- $-: \mathbb{R}^{m \times n} \times \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$

For example, addition of two 3x2 matrices a and b:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \\ a_{31} + b_{31} & a_{32} + b_{32} \end{bmatrix}$$

Matrices must have the same dimensions to be added or subtracted.

# Scaling by a constant

- Multiply a matrix by a constant (scalar) value.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \alpha = \begin{bmatrix} a_{11}\alpha & a_{12}\alpha \\ a_{21}\alpha & a_{22}\alpha \\ a_{31}\alpha & a_{32}\alpha \end{bmatrix}$$

# Matrix Product

- Done by taking the dot product between each row of matrix A and each column of matrix B to produce matrix C.

$$\begin{array}{c} \overrightarrow{r_1} \rightarrow \\ \overrightarrow{r_2} \rightarrow \end{array} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{array}{c} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \\ \begin{array}{cc} \uparrow & \qquad \uparrow \\ \overrightarrow{c_1} & \qquad \overrightarrow{c_2} \end{array} \end{array} = \begin{bmatrix} \overrightarrow{r_1} \cdot \overrightarrow{c_1} & \overrightarrow{r_1} \cdot \overrightarrow{c_2} \\ \overrightarrow{r_2} \cdot \overrightarrow{c_1} & \overrightarrow{r_2} \cdot \overrightarrow{c_2} \end{bmatrix}$$

The rows of A must have the same dimension as the columns of B, and the rows of B must have the same dimension as the columns of C.

# Matrix Product rules

- The rows of A must have the same dimension as the columns of B, and the rows of B must have the same dimension as the columns of C.

- Given two matrices $A \in \mathbb{R}^{m \times n}$ and $\mathrm{B} \in \mathbb{R}^{n \times k}$, the product AB is an $m \times k$ matrix.

- Matrix multiplication is not a commutative operation: $AB \neq BA$

# Matrix-vector product

- Product between matrix A and vector $\vec{v}$. We need to view the vector as a column matrix.
- Dot product of $\vec{v}$ with every row of A.
- The number of columns in A equals the number of columns in $\vec{v}$.

For example:

$$\begin{array}{l} \vec{r_1} \rightarrow \\ \vec{r_2} \rightarrow \end{array} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \overrightarrow{r_1} \cdot \vec{v} \\ \overrightarrow{r_2} \cdot \vec{v} \end{bmatrix}$$

# Matrix Inverse

- The formula for the inverse of a square matrix:

$$A^{-1} = \frac{1}{|A|} \cdot Adj\ A$$

- A is the square matrix (n x n).
- Adj(A) is the adjoint matrix of A.
- $|A|$ is the determinant of A.

- Note: the matrix should be square, and the determinant should not be equal to zero.

# Matrix Trace

- The sum of elements on the main diagonal from the upper left to the lower right of A.
- Can only be for a square matrix.
- Example:

$$\begin{bmatrix} \boldsymbol{a_{11}} & a_{12} & a_{13} \\ a_{21} & \boldsymbol{a_{22}} & a_{23} \\ a_{31} & a_{32} & \boldsymbol{a_{33}} \end{bmatrix}$$

# Matrix Determinant

- The determinant of matrix A, is an operation that give us useful information about the linear independence of the rows of the matrix.
- The determinant of a triangular matrix is the product of its diagonal entries.
- The determinant of 2x2 matrix:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

# Matrix Determinant Example

- The determinant of 3x3 matrix:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = a_{11}a_{22}a_{23} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}$$

- Pattern:

$$\begin{bmatrix} a_{11} & \mathbf{X} & & \\ & & a_{22} & a_{23} \\ & & a_{32} & a_{33} \end{bmatrix} - \begin{bmatrix} & a_{12} & \\ & \mathbf{X} & \\ a_{21} & & a_{23} \\ a_{31} & & a_{33} \end{bmatrix} + \begin{bmatrix} & & \mathbf{X} & a_{13} \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & & \end{bmatrix}$$

# Homework

Develop Python code for:

- Vector: dot product and cross product.
- Matrix: Addition, substraction, matrix product, matrix-vector product.

Rules:

- Use only Python (no libraries).
- Put in comments the complexity of the code (O notation).
- Use Numpy as well and compare the time it takes to run with both of them.
- Send it to us by mail in a zip file (no .py code).

# Homework function descriptions

1. **vector_dot_product(v1, v2)**: Computes the dot product of two vectors.
2. **vector_cross_product(v1, v2)**: Computes the cross product of two 3D vectors.
3. **matrix_addition(A, B)**: Adds two matrices of the same size.
4. **matrix_subtraction(A, B)**: Subtracts matrix B from matrix A, element-wise.
5. **matrix_product(A, B)**: Multiplies two matrices.
6. **matrix_vector_product(A, v)**: Multiplies a matrix by a vector.

# Use case

- Logistic Regression algorithm:

The training process using gradient descent follows these steps:

1. **Initialize the weights**: Start with random values for $w_0, w_1, w_2$.
2. **For each epoch** (iteration over the entire dataset):
   1. For each training example ($x_1, x_2$,y) :
      1. Compute $z = w_0 + w_1 x_1 + w_2 x_2$
      2. Apply the sigmoid function to get the predicted probability $\hat{y} = \sigma(z)$.
      3. Compute the loss using binary cross-entropy.
      4. Compute the gradients for each weight $w_0, w_1, w_2$ .
      5. Update the weights using gradient descent.
   2. After the epoch, evaluate the performance of the model (optional).
3. **Repeat** until the loss converges (i.e., stops changing significantly).