

Cours Algebre linéaire

Novembre 2024

Contents

1	Introduction	4
1.1	Qu'est ce que l'algèbre linéaire ?	4
2	Théorie des ensembles	6
2.1	Groupes	6
2.2	Anneaux	7
2.3	Anneaux intègres	7
2.4	Corps	8
2.5	Corps algébriquement clos	8
2.6	Espace des matrices	8
3	Vecteurs	9
3.1	Représentation géométrique	9
3.2	Addition de vecteurs	10
3.3	Multiplication d'un vecteur par un scalaire	11
3.4	Vecteurs Colinéaires	11
3.5	Produit scalaire	12
3.6	Norme d'un vecteur	13
3.7	Cosinus de l'Angle entre Deux Vecteurs	14
3.8	Produit vectoriel en dimension 3	15
4	Matrices	16
4.1	Lien entre Matrices et Vecteurs	16
4.2	Interprétation géométrique	17
4.3	Interprétation en machine learning	19
4.4	Matrices Particulières	20
4.5	Addition et soustraction de matrices	26
4.6	Multiplication de 2 Matrices	27
4.7	Transposée d'une Matrice	30
4.8	Trace d'une Matrice	31
4.9	Rang d'une Matrice	33
4.10	Matrice orthogonale	34

5	Espaces vectoriels	35
5.1	Définition et exemple	35
5.2	Sous-espace vectoriels (SEV)	35
5.3	Somme directe de sous-espaces vectoriels	36
5.4	Notions affines et vectorielles	38
5.5	Combinaisons linéaires	38
5.6	Familles libres (linéairement indépendantes)	39
5.7	Familles génératrices	39
5.8	Bases et dimension	40
6	Applications Linéaires	42
6.1	Définition	42
6.2	Exemples	42
6.3	Propriétés	42
6.4	Isomorphisme	42
6.5	Noyau et Image	43
6.6	Matrice d'une Application Linéaire	44
6.7	Matrices de Passage	45
6.8	Composition et Inverse	46
6.9	Exemple d'Application en Machine Learning	46
7	Déterminant d'une Matrice	47
7.1	Définition	47
7.2	Calcul du Déterminant	47
7.3	Propriétés des Opérations sur les Lignes et Colonnes	51
7.4	Matrices Triangulaires et Diagonales	53
7.5	Propriétés du Déterminant	54
7.6	Interprétation Géométrique du Déterminant	54
7.7	Pseudo-Déterminant	57
7.8	Interprétation du Déterminant en Machine Learning	58
8	Inverse d'une matrice	59
8.1	Vers la résolution des équations matricielles	59
8.2	L'Inverse de la Matrice	59
8.3	Types d'Inverses et Conditions d'Invertibilité	59
8.4	Calcul de l'inverse	60
8.5	L'Inverse complet est Unique	67
8.6	Interprétation Géométrique de l'Inverse	68
8.7	Résumé des situations	69
8.8	Application : Résolution d'une Équation Matricielle	70
9	Éléments propres et polynôme caractéristique	74
9.1	Vecteurs et valeurs propres	74
9.2	Polynôme caractéristique	74
9.3	Calcul des valeurs propres et des vecteurs propres	76
9.4	Propriétés	77

9.5	Multiplicité des valeurs propres	78
9.6	Lien avec la trace et le déterminant	79
9.7	Matrice non inversible et valeur propre nulle	82
10	Diagonalisation d'une matrice	83
10.1	Définition	83
10.2	Interprétation géométrique	84
10.3	Conditions de diagonalisation	84
10.4	Procédure de diagonalisation	84
10.5	Cas des matrices non diagonalisables	85
10.6	Exemple	85
10.7	Remarques	87
11	Trigonalisation d'une matrice	89
11.1	Définition	89
11.2	Conditions de trigonalisation	89
11.3	Étapes du processus de trigonalisation	89
11.4	Méthodes avancées : Forme de Jordan	90
11.5	Lien avec la décomposition de Dunford	90
12	Décomposition en valeurs singulières	91
12.1	Introduction à la SVD	91
12.2	Construction de matrices symétriques associées	92
12.3	Calcul des valeurs et vecteurs singuliers	92
12.4	Visualisation géométrique de la SVD	94
12.5	Approximation de rang faible et applications	95
12.6	Lien entre la SVD et la Réduction de Dimension	98
12.7	La SVD et le Pseudo-Inverse de Moore-Penrose	100
12.8	Lien avec le chapitre sur le déterminant	102
12.9	Remarque avec NumPy	104

1 Introduction

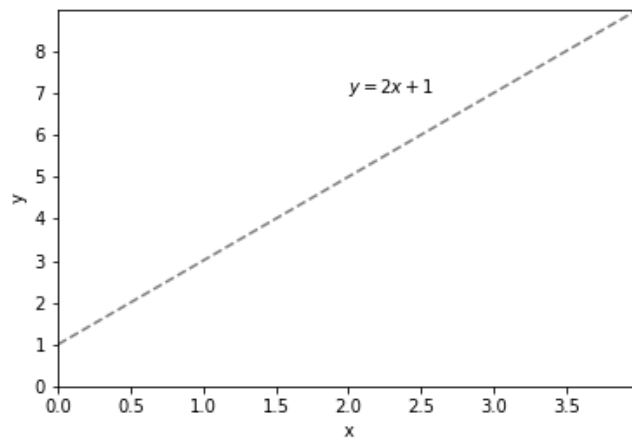
Quand on débute en algèbre linéaire, il est courant de se sentir déstabilisé par l'aspect abstrait et inédit de ce langage. Ne vous découragez pas ! Au fil de votre progression dans ce cours, les concepts abordés auparavant vous sembleront de plus en plus familiers. À la fin, l'algèbre linéaire deviendra comme une langue étrangère : vous la maîtriserez sans même vous rappeler comment vous l'avez apprise.

1.1 Qu'est ce que l'algèbre linéaire ?

L'algèbre linéaire est une branche des mathématiques, souvent abstraite et difficile aux novices, mais qui trouve son application dans quasiment tous les domaines scientifiques et techniques. Quand on désire comprendre en profondeur ce qu'est le Machine Learning, il est indispensable de maîtriser les bases de l'algèbre linéaire et avoir une bonne appréciation de ces différents piliers : les espace vectoriels, les décompositions matricielles, etc.

D'un point de vue mathématique, l'algèbre linéaire est l'étude des lignes, des plans, et de tout espace vectoriel responsables de transformations linéaires.

Commençons par un exemple simple. L'équation $y = 2x + 1$ peut être vue non seulement comme une simple relation entre x et y , mais aussi comme une façon de décrire une ligne dans le plan 2D. En variant x , on génère différentes valeurs de y qui, ensemble, définissent cette ligne.



Plutôt que de voir chaque point individuellement, nous pouvons représenter l'ensemble des points de la ligne comme une transformation. Supposons que nous regroupions toutes les valeurs x dans un vecteur X (un ensemble de valeurs

x sous forme de colonne), alors l'ensemble correspondant des valeurs y pourrait être représenté comme un autre vecteur, Y .

Formulation matricielle

Pour faire cette correspondance entre les valeurs x et y , on utilise une matrice de transformation M , qui contient les coefficients de la relation initiale. Ici, la transformation $Y = M \cdot X$ est donnée par :

$$M = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Ainsi, M agit comme un multiplicateur, transformant chaque valeur x en une valeur y via la relation vectorielle.

Le même principe peut être étendu aux équations de plusieurs variables, comme $y = 2x_1 + 0.5x_2 - 2$, où chaque variable x_1, x_2 contribue à la valeur de y . Dans ce cas, M deviendrait une matrice qui décrit la relation entre les différents x et y . On pourrait alors passer des points dans un espace de plusieurs dimensions (ou vecteur de plusieurs variables X) vers un espace de résultats Y par une matrice plus complexe.

En somme, la matrice M traduit comment les valeurs d'entrée X sont modifiées en sortie Y par une transformation linéaire.

2 Théorie des ensembles

Ce chapitre sert principalement à établir les fondements et le cadre pour les concepts mathématiques plus avancés que nous aborderons par la suite. Si vous débutez en mathématiques, il n'est pas nécessaire de comprendre en détail ce contenu dès maintenant. Vous pouvez le parcourir pour vous familiariser avec les notions de base, puis y revenir ultérieurement lorsque vous serez plus à l'aise avec les concepts fondamentaux.

2.1 Groupes

Un **groupe** (G, \star) est un ensemble G muni d'une loi de composition interne \star satisfaisant aux propriétés suivantes :

1. **Associativité** : $\forall a, b, c \in G, (a \star b) \star c = a \star (b \star c)$
2. **Élément neutre** : Il existe un élément $e \in G$ tel que $\forall a \in G, e \star a = a \star e = a$
3. **Inverse** : Pour chaque élément $a \in G$, il existe un élément $a^{-1} \in G$ tel que $a \star a^{-1} = a^{-1} \star a = e$

Si de plus la loi \star est commutative, c'est-à-dire :

$$\forall a, b \in G, a \star b = b \star a$$

alors (G, \star) est appelé un **groupe commutatif** ou **groupe abélien**.

Exemples

- L'ensemble des entiers relatifs \mathbb{Z} avec l'addition $(+)$ forme un groupe commutatif.
- L'ensemble des nombres réels \mathbb{R} avec l'addition $(+)$ forme un groupe commutatif.
- L'ensemble des nombres non nuls $\mathbb{R}^* = \mathbb{R} \setminus \{0\}$ avec la multiplication (\cdot) forme un groupe commutatif.

Exemples concrets

- **Heure sur une horloge** : L'ensemble des heures sur une horloge (0 à 11) avec l'opération d'addition modulo 12 forme un groupe. Par exemple, $10 + 3 = 1$ (on revient à 1 heure après minuit). L'élément neutre est 0, et chaque heure a un inverse pour revenir à 0.
- **Rotation d'objets** : Les rotations possibles d'un carré autour de son centre ($0^\circ, 90^\circ, 180^\circ, 270^\circ$) forment un groupe avec la composition de rotations. Par exemple, tourner de 90° deux fois équivaut à tourner de 180° . L'élément neutre est la rotation de 0° , et chaque rotation a un inverse (par exemple, 90° et 270°).

2.2 Anneaux

Un **anneau** $(A, +, \cdot)$ est un ensemble A muni de deux lois de composition interne, l'addition $+$ et la multiplication \cdot , satisfaisant aux propriétés suivantes :

1. $(A, +)$ est un groupe commutatif.
2. **Associativité de la multiplication** : $\forall a, b, c \in A, (a \cdot b) \cdot c = a \cdot (b \cdot c)$
3. **Distributivité** : $\forall a, b, c \in A, a \cdot (b + c) = a \cdot b + a \cdot c$ et $(a + b) \cdot c = a \cdot c + b \cdot c$

Si la multiplication est aussi commutative, l'anneau est dit **commutatif**.

Exemples concrets

- **Les entiers \mathbb{Z}** : Les entiers avec l'addition et la multiplication forment un anneau commutatif. Ces opérations suivent les règles de base de l'arithmétique.
- **Horloge de 12 heures avec addition et multiplication modulo 12** : En utilisant les heures comme éléments, cet ensemble est un anneau, car les opérations d'addition et de multiplication respectent la distributivité.

2.3 Anneaux intègres

Un **anneau intègre** est un anneau commutatif non nul $(A, +, \cdot)$ dans lequel le produit de deux éléments non nuls est non nul, c'est-à-dire :

$$\forall a, b \in A, a \cdot b = 0 \Rightarrow a = 0 \text{ ou } b = 0$$

Exemples concrets

- **Entiers \mathbb{Z}** : Les entiers forment un anneau intègre car le produit de deux entiers non nuls est toujours non nul.

2.4 Corps

Un **corps** $(\mathbb{K}, +, \cdot)$ est un ensemble \mathbb{K} muni de deux opérations, l'addition et la multiplication, satisfaisant aux propriétés d'un anneau intègre, et où tout élément non nul possède un inverse pour la multiplication.

Exemples concrets

- **Les nombres rationnels \mathbb{Q}** : Les fractions forment un corps, car on peut additionner, multiplier, et inverser (sauf le zéro) chaque élément.
- **Les nombres réels \mathbb{R} et complexes \mathbb{C}** : Ce sont des corps, car chaque élément non nul a un inverse.

2.5 Corps algébriquement clos

Un **corps algébriquement clos** est un corps \mathbb{K} dans lequel tout polynôme non constant à coefficients dans \mathbb{K} admet au moins une racine dans \mathbb{K} . Autrement dit, tout polynôme non constant de $\mathbb{K}[X]$ peut être factorisé en produit de polynômes de degré 1 à coefficients dans \mathbb{K} .

Remarque : Dans le cadre de l'algèbre linéaire, travailler sur un corps algébriquement clos comme \mathbb{C} garantit que le polynôme caractéristique de toute matrice carrée est scindé, c'est-à-dire qu'il peut être factorisé en produit de facteurs du premier degré. Cela aura des implications pour des concepts tels que la diagonalisation que nous verrons par la suite.

2.6 Espace des matrices

Soit \mathbb{K} un corps (par exemple \mathbb{R} ou \mathbb{C}). On note $\mathcal{M}_{m,n}(\mathbb{K})$ l'ensemble des matrices à m lignes et n colonnes, avec des coefficients dans \mathbb{K} . Une matrice $A \in \mathcal{M}_{m,n}(\mathbb{K})$ est un tableau de la forme :

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

où chaque $a_{ij} \in \mathbb{K}$.

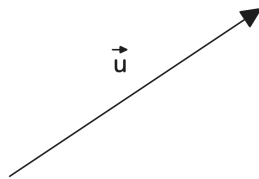
Exemples

- **Matrices de transformation** : Les matrices peuvent être utilisées pour décrire des transformations comme la rotation ou la translation.
- **Calculs de probabilités** : Dans les probabilités, les matrices servent à organiser et calculer des transitions entre états dans les modèles de Markov.

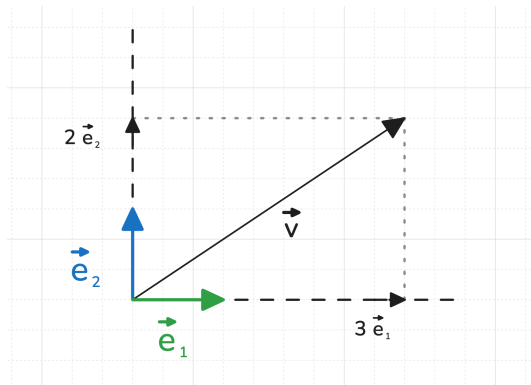
3 Vecteurs

3.1 Représentation géométrique

Un vecteur est un objet mathématique qui possède à la fois une magnitude (ou longueur) et une direction. Il est souvent représenté par une flèche dont la direction indique le sens du vecteur et la longueur indique sa magnitude. On note $||\vec{v}||$ la longueur d'un vecteur, que l'on appelle sa norme, et qui est un réel positif ou nul. Si sa norme vaut 1 alors on dit que c'est un vecteur **unitaire**.



On note également $\vec{0}$ le vecteur de longueur nulle. Ce vecteur ne possède aucune direction. Un vecteur s'exprime avec ses coordonnées, qui correspondent au déplacement nécessaire à effectuer entre l'origine du vecteur et l'arrivée, c'est à dire suivant l'axe x puis l'axe y si nous sommes dans un espace de dimension 2 comme \mathbb{R}^2 .



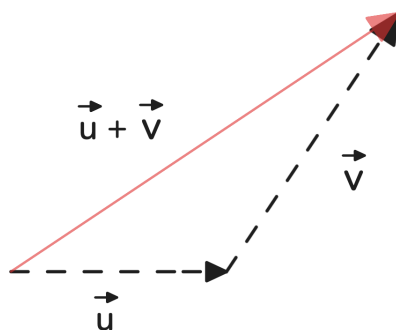
Ici $\vec{v} = 3\vec{e}_1 + 2\vec{e}_2 = (3, 2)$

```
import numpy as np

# Définition d'un vecteur en 2D
vecteur = np.array([3, 2])
print("Vecteur :", vecteur)
```

3.2 Addition de vecteurs

Pour additionner des vecteurs, on fait la somme composante par composante. Pour 2 vecteurs de \mathbb{R}^n : $\vec{u} = (u_1, \dots, u_n)$ et $\vec{v} = (v_1, \dots, v_n)$, alors $\vec{u} + \vec{v} = (u_1 + v_1, \dots, u_n + v_n)$. Graphiquement, cela peut être représenté par le placement bout à bout de chacun des vecteurs dans l'ordre dans lequel ils sont écrit dans la somme.

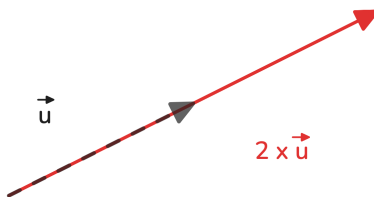


```
import numpy as np

# Addition de deux vecteurs en 2D
vecteur_u = np.array([1, 2])
vecteur_v = np.array([3, 4])
vecteur_somme = vecteur_u + vecteur_v
print("Somme des vecteurs :", vecteur_somme)
```

3.3 Multiplication d'un vecteur par un scalaire

Multiplier un vecteur par un scalaire consiste à multiplier chacune de ses composantes par ce scalaire, ce qui a pour effet de changer la longueur du vecteur sans modifier sa direction. Par exemple, si c est un scalaire et $\vec{u} = (u_1, \dots, u_n)$ un vecteur de \mathbb{R}^n , alors $c * \vec{u} = (c * u_1, \dots, c * u_n)$.



```
import numpy as np

# Multiplication d'un vecteur par un scalaire
scalaire = 3
vecteur_u = np.array([2, -5])
vecteur_resultat = scalaire * vecteur_u
print("Vecteur résultant :", vecteur_resultat)
```

3.4 Vecteurs Colinéaires

Deux vecteurs sont dits **colinéaires** s'ils sont alignés sur la même droite, c'est-à-dire si l'un est un multiple scalaire de l'autre. Formellement, deux vecteurs \vec{u} et \vec{v} dans \mathbb{R}^n sont colinéaires s'il existe un scalaire $c \in \mathbb{R}$ tel que :

$$\vec{v} = c \cdot \vec{u}$$

Par exemple :

$$a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}; \quad b = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

Les deux vecteurs a et b sont colinéaires car $b = 2 \times a$.

3.5 Produit scalaire

Le produit scalaire de deux vecteurs \vec{u} et \vec{v} est une mesure de leur alignement et est donné par $\vec{u} \cdot \vec{v} = \sum_i u_i v_i$. Ce calcul donne un scalaire qui peut être utilisé pour déterminer l'angle entre deux vecteurs.

Exemple : Si $\vec{u} = (1, 2, 3)$ et $\vec{v} = (4, -5, 6)$,

$$\vec{u} \cdot \vec{v} = 1 \times 4 + 2 \times -5 + 3 \times 6 = 4 - 10 + 18 = 12.$$

Lorsque le produit scalaire de 2 vecteurs est nul alors les vecteurs sont orthogonaux.

```
import numpy as np

# Produit scalaire de deux vecteurs en 3D
vecteur_u = np.array([1, 2, 3])
vecteur_v = np.array([4, -5, 6])
produit_scalaire = np.dot(vecteur_u, vecteur_v)
print("Produit scalaire :", produit_scalaire)
```

3.6 Norme d'un vecteur

La **norme** d'un vecteur $\vec{v} = (v_1, v_2, \dots, v_n)$ dans \mathbb{R}^n est une mesure de la longueur (ou de la magnitude) de ce vecteur. Elle est définie par :

$$\|\vec{v}\| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$$

Exemples

- Pour un vecteur $\vec{v} = (3, 4)$ dans \mathbb{R}^2 , la norme est calculée comme suit :

$$\|\vec{v}\| = \sqrt{3^2 + 4^2} = \sqrt{9 + 16} = \sqrt{25} = 5$$

- Pour un vecteur $\vec{w} = (1, 2, 2)$ dans \mathbb{R}^3 , la norme est calculée comme suit :

$$\|\vec{w}\| = \sqrt{1^2 + 2^2 + 2^2} = \sqrt{1 + 4 + 4} = \sqrt{9} = 3$$

```
import numpy as np

# Calcul de la norme d'un vecteur en 2D
vecteur = np.array([3, 4])
norme = np.linalg.norm(vecteur)
print("Norme du vecteur 2D :", norme)

# Calcul de la norme d'un vecteur en 3D
vecteur = np.array([1, 2, 2])
norme = np.linalg.norm(vecteur)
print("Norme du vecteur 3D :", norme)
```

Propriétés de la norme

1. **Positivité** : Pour tout vecteur $\vec{v} \in \mathbb{R}^n$, $\|\vec{v}\| \geq 0$ et $\|\vec{v}\| = 0$ si et seulement si $\vec{v} = \vec{0}$.
2. **Homogénéité** : Pour tout scalaire $\alpha \in \mathbb{R}$ et tout vecteur $\vec{v} \in \mathbb{R}^n$, $\|\alpha\vec{v}\| = |\alpha| \cdot \|\vec{v}\|$.
3. **Inégalité triangulaire** : Pour tous vecteurs $\vec{u}, \vec{v} \in \mathbb{R}^n$, $\|\vec{u} + \vec{v}\| \leq \|\vec{u}\| + \|\vec{v}\|$.

3.7 Cosinus de l'Angle entre Deux Vecteurs

Le produit scalaire est lié au cosinus de l'angle θ entre les deux vecteurs :

$$\vec{u} \cdot \vec{v} = \|\vec{u}\| \|\vec{v}\| \cos(\theta).$$

Cela permet de définir la similarité cosinus, largement utilisée en machine learning pour mesurer la similarité entre deux vecteurs (par exemple, dans le traitement du langage naturel pour comparer des documents ou des phrases).

La similarité cosinus entre deux vecteurs \vec{u} et \vec{v} est définie comme :

$$\text{Similitude}(\vec{u}, \vec{v}) = \cos(\theta) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \|\vec{v}\|}$$

Cette mesure varie entre -1 et 1 et est largement utilisée pour comparer des documents en traitement du langage naturel.

Exemple : Soit $\vec{u} = (1, 0, 1)$ et $\vec{v} = (0, 1, 1)$,

$$\vec{u} \cdot \vec{v} = 1 * 0 + 0 * 1 + 1 * 1 = 0 + 0 + 1 = 1$$

$$\|\vec{u}\| = \sqrt{1^2 + 0^2 + 1^2} = \sqrt{2}$$

$$\|\vec{v}\| = \sqrt{0^2 + 1^2 + 1^2} = \sqrt{2}$$

$$\cos(\theta) = \frac{1}{\sqrt{2} \times \sqrt{2}} = \frac{1}{2}$$

Une similarité de 0.5 indique que les vecteurs sont relativement similaires.

```
import numpy as np

# Calcul de la similarité cosinus entre deux vecteurs
vecteur_u = np.array([1, 0, 1])
vecteur_v = np.array([0, 1, 1])

# Produit scalaire
produit_scalaire = np.dot(vecteur_u, vecteur_v)

# Normes des vecteurs
norme_u = np.linalg.norm(vecteur_u)
norme_v = np.linalg.norm(vecteur_v)

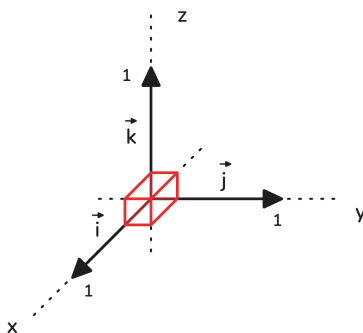
# Calcul du cosinus de l'angle
cos_theta = produit_scalaire / (norme_u * norme_v)
print("Similarité cosinus :", cos_theta)
```

3.8 Produit vectoriel en dimension 3

Le produit vectoriel de deux vecteurs $\vec{u} = (u_1, u_2, u_3)$ et $\vec{v} = (v_1, v_2, v_3)$ est un vecteur qui est perpendiculaire à la fois à \vec{u} et à \vec{v} . Sa magnitude est égale à l'aire du parallélogramme formé par \vec{u} et \vec{v} . Pour des vecteurs dans \mathbb{R}^3 :

$$\vec{u} \times \vec{v} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix}.$$

Où \hat{i} , \hat{j} et \hat{k} sont les vecteurs unitaires suivant les axes x, y et z de \mathbb{R}^3 (que l'on appellera base de \mathbb{R}^3 par la suite).



Exemple : Avec $\vec{u} = (1, 2, 3)$ et $\vec{v} = (4, -5, 6)$,

$$\begin{aligned} \vec{u} \times \vec{v} &= \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ 1 & 2 & 3 \\ 4 & -5 & 6 \end{vmatrix} \\ &= \hat{i}(2 \cdot 6 - 3 \cdot -5) - \hat{j}(1 \cdot 6 - 3 \cdot 4) + \hat{k}(1 \cdot -5 - 2 \cdot 4) \\ &= \hat{i}(12 + 15) - \hat{j}(6 - 12) + \hat{k}(-5 - 8) \\ &= 27\hat{i} + 6\hat{j} - 13\hat{k}. \end{aligned}$$

```
import numpy as np

# Produit vectoriel de deux vecteurs en 3D
vecteur_u = np.array([1, 2, 3])
vecteur_v = np.array([4, -5, 6])

produit_vectoriel = np.cross(vecteur_u, vecteur_v)
print("Produit vectoriel :", produit_vectoriel)
```

4 Matrices

4.1 Lien entre Matrices et Vecteurs

Une matrice est un tableau de nombres organisé en lignes et en colonnes. Elle peut être vue comme une collection de vecteurs disposés soit en lignes, soit en colonnes.

Vecteur comme Matrice : Un vecteur de dimension n peut être considéré comme une matrice de taille $n \times 1$ (vecteur colonne) ou $1 \times n$ (vecteur ligne). Par exemple, un vecteur $\vec{v} = (v_1, v_2, v_3)$ peut être représenté comme :

Vecteur colonne :

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Vecteur ligne :

$$\vec{v} = [v_1 \quad v_2 \quad v_3]$$

Matrice comme Collection de Vecteurs : Une matrice peut être vue comme une collection de vecteurs colonnes ou de vecteurs lignes. Par exemple, la matrice A suivante est composée de vecteurs colonnes $\vec{c}_1, \vec{c}_2, \vec{c}_3$ ou de vecteurs lignes \vec{r}_1, \vec{r}_2 :

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} = [\vec{c}_1 \quad \vec{c}_2 \quad \vec{c}_3] = \begin{bmatrix} \vec{r}_1 \\ \vec{r}_2 \end{bmatrix}$$

Où :

- $\vec{c}_1 = \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}, \vec{c}_2 = \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix}, \vec{c}_3 = \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}$
- $\vec{r}_1 = [a_{11} \quad a_{12} \quad a_{13}], \vec{r}_2 = [a_{21} \quad a_{22} \quad a_{23}]$

```
import numpy as np

# Vecteur colonne
v_colonne = np.array([[1],
                      [2],
                      [3]])
print("Vecteur colonne :\n", v_colonne)

# Vecteur ligne
v_ligne = np.array([1, 2, 3])
print("Vecteur ligne :", v_ligne)
```


4.2 Interprétation géométrique

Une matrice peut être vue comme une transformation linéaire des axes d'un repère. En géométrie, appliquer une matrice à un vecteur revient à transformer ce vecteur en modifiant sa direction et sa longueur selon les colonnes de la matrice, qui représentent les nouveaux axes du repère après transformation.

Note: repère orthonormé

Un repère orthonormé est un système de coordonnées dans lequel les axes sont orthogonaux (formant des angles de 90° entre eux) et chaque axe est de longueur unitaire (de norme 1). Cela simplifie grandement les calculs géométriques et algébriques, comme la mesure des distances ou des angles.

Considérons un repère orthonormé (O, \vec{i}, \vec{j}) . Si l'on applique la matrice

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

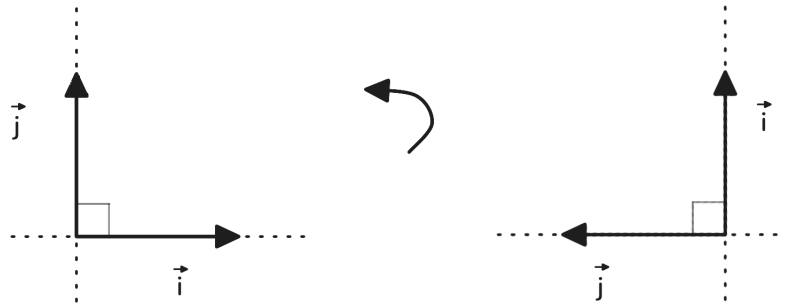
alors le vecteur \vec{i} est transformé en

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

et le vecteur \vec{j} est transformé en

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

Ce qui montre que le vecteur \vec{i} a à présent comme coordonnées $(0, 1)$ et le vecteur \vec{j} a à présent comme coordonnées $(-1, 0)$, illustrant une rotation de 90 degrés dans le sens inverse des aiguilles d'une montre, comme illustré dans l'image suivante.



L'opérateur @ représente le produit matriciel, nous le verrons juste après.

```
import numpy as np

# Matrice de rotation de 90 degrés
M = np.array([[0, -1],
              [1,  0]])

# Vecteurs de base
i = np.array([1, 0])
j = np.array([0, 1])

# Transformation des vecteurs
i_transforme = M @ i
j_transforme = M @ j

print("i transformé :", i_transforme)
print("j transformé :", j_transforme)
```

4.3 Interprétation en machine learning

En machine learning, une matrice peut être interprétée comme un ensemble de données où chaque ligne représente un individu ou un exemple de données, et chaque colonne représente une caractéristique (ou *feature*) de ces données. Cette vue matricielle permet de manipuler les données efficacement pour des tâches d'analyse, de modélisation et d'apprentissage automatique.

Considérons un exemple simple où nous avons un ensemble de données contenant des informations sur trois personnes. Chaque personne est décrite par trois caractéristiques : l'âge, la taille en centimètres, et le poids en kilogrammes. La matrice des données pourrait être représentée comme suit :

$$\mathbf{X} = \begin{bmatrix} 25 & 180 & 75 \\ 30 & 175 & 70 \\ 22 & 160 & 55 \end{bmatrix}$$

Dans cet exemple :

- 3 individus (lignes) :

- Individu 1 : 25 ans, 180 cm, 75 kg
- Individu 2 : 30 ans, 175 cm, 70 kg
- Individu 3 : 22 ans, 160 cm, 55 kg

- 3 caractéristiques (colonnes) :

- Âge
- Taille (cm)
- Poids (kg)

Pour une matrice de données \mathbf{X} de dimension $m \times n$, où m est le nombre d'individus et n est le nombre de caractéristiques, chaque entrée \mathbf{X}_{ij} représente la valeur de la caractéristique j pour l'individu i .

Application pratique En apprentissage automatique, cette représentation matricielle permet d'appliquer des algorithmes de manière efficace. Par exemple, dans une régression linéaire, nous cherchons à établir une relation entre les caractéristiques (features) et une variable cible (target). Si nous ajoutons une colonne supplémentaire pour la variable cible, nous pourrions avoir :

$$\mathbf{X} = \begin{bmatrix} \text{Âge} & \text{Taille (cm)} & \text{Poids (kg)} & \text{Variable cible (ex: IMC)} \\ 25 & 180 & 75 & 23.1 \\ 30 & 175 & 70 & 22.9 \\ 22 & 160 & 55 & 21.5 \end{bmatrix}$$

Cette structure matricielle facilite l'utilisation de bibliothèques de machine learning, telles que Scikit-learn, qui manipulent les données en formats matriciels pour entraîner des modèles, effectuer des prédictions et évaluer la performance.

Remarque : Lorsque vous utilisez un DataFrame de pandas pour entraîner des modèles avec scikit-learn, celui-ci est converti en matrice NumPy en interne.

4.4 Matrices Particulières

Il existe plusieurs types de matrices particulières en algèbre linéaire, chacune ayant des propriétés spécifiques et des applications utiles.

4.4.1 Matrice Identité

Une matrice identité, notée I_n , est une matrice carrée de taille $n \times n$ dont les éléments diagonaux sont égaux à 1 et tous les autres éléments sont égaux à 0. Elle joue un rôle similaire à celui de l'élément neutre pour la multiplication :

$$I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Pour toute matrice A de dimension $n \times n$, on a $AI_n = I_nA = A$.

```
import numpy as np

# Matrice identité de taille 3x3
I = np.eye(3)
print("Matrice identité :\n", I)
```

4.4.2 Matrice Nulle

Une matrice nulle, notée $0_{m,n}$, est une matrice de taille $m \times n$ dont tous les éléments sont égaux à 0 :

$$0_{m,n} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

```
import numpy as np

# Matrice nulle de taille 2x3
zero_matrix = np.zeros((2, 3))
print("Matrice nulle :\n", zero_matrix)
```

4.4.3 Matrice Colonne et Matrice Ligne

Matrice Colonne Une matrice colonne est une matrice de taille $m \times 1$, c'est-à-dire qu'elle a une seule colonne :

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$$

Matrice Ligne Une matrice ligne est une matrice de taille $1 \times n$, c'est-à-dire qu'elle a une seule ligne :

$$[a_1 \quad a_2 \quad \cdots \quad a_n]$$

4.4.4 Matrice Carrée

Une matrice carrée est une matrice de taille $n \times n$, c'est-à-dire qu'elle a autant de lignes que de colonnes :

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

4.4.5 Matrice Diagonale

Une matrice diagonale est une matrice carrée dans laquelle tous les éléments hors diagonale sont nuls :

$$D = \begin{bmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{nn} \end{bmatrix}$$

On la note aussi $D = \text{diag}(d_{11}, d_{22}, d_{nn})$

```
import numpy as np

# Matrice diagonale avec des valeurs spécifiques
D = np.diag([1, 2, 3])
print("Matrice diagonale :\n", D)
```

4.4.6 Matrice Triangulaire

Matrice Triangulaire Supérieure Une matrice triangulaire supérieure est une matrice carrée dans laquelle tous les éléments en dessous de la diagonale sont nuls :

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

```
import numpy as np

# Matrice aléatoire
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# Matrice triangulaire supérieure
U = np.triu(A)
print("Matrice triangulaire supérieure :\n", U)
```

Matrice Triangulaire Inférieure Une matrice triangulaire inférieure est une matrice carrée dans laquelle tous les éléments au-dessus de la diagonale sont nuls :

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}$$

```
import numpy as np

# Matrice aléatoire
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# Matrice triangulaire inférieure
L = np.tril(A)
print("Matrice triangulaire inférieure :\n", L)
```

4.4.7 Matrice Symétrique

Une matrice symétrique est une matrice carrée qui est égale à sa transposée, c'est-à-dire $A = A^T$. Cela implique que $a_{ij} = a_{ji}$ pour tous les i et j :

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

```
import numpy as np

# Matrice symétrique
A = np.array([[1, 2, 3],
              [2, 5, 6],
              [3, 6, 9]])

# Vérification que A est symétrique
print("Matrice A :\n", A)
print("Transposée de A :\n", A.T)
print("A est symétrique :", np.array_equal(A, A.T))
```

4.4.8 Matrices Rectangulaires

Une matrice rectangulaire est une matrice dont le nombre de lignes m est différent du nombre de colonnes n . Ces matrices sont essentielles car elles permettent de transformer des vecteurs d'un espace de dimension n en vecteurs d'un espace de dimension m .

Une matrice M de taille $m \times n$ réalise une transformation linéaire qui prend un vecteur de \mathbb{R}^n et le transforme en un vecteur de \mathbb{R}^m .

- Si $n > m$, la matrice M **réduit** la dimension de l'espace en supprimant des composantes du vecteur original.
- Si $n < m$, la matrice M **augmente** la dimension de l'espace en ajoutant de nouvelles composantes (généralement nulles ou définies par la transformation).

Exemple :

Considérons la matrice rectangulaire suivante :

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Cette matrice est de taille 2×3 , elle prend donc des vecteurs de \mathbb{R}^3 et les transforme en vecteurs de \mathbb{R}^2 . Elle peut être vue comme une matrice diagonale rectangulaire qui conserve uniquement les deux premières composantes du vecteur d'entrée, supprimant ainsi la troisième composante.

Application de la matrice :

Prenons un vecteur $v \in \mathbb{R}^3$:

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Alors,

$$Mv = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

La transformation M prend le vecteur v et produit un vecteur de \mathbb{R}^2 en conservant les deux premières composantes et en supprimant la troisième.


```

import numpy as np

# Matrice rectangulaire M de taille 2x3
M = np.array([[1, 0, 0],
              [0, 1, 0]])

# Vecteur v dans R^3
v = np.array([4, -2, 5])

# Transformation de v par M
Mv = M.dot(v)

print("Matrice M :\n", M)
print("Vecteur v :", v)
print("M v :", Mv)

```

De même, si nous avons une matrice de taille 3×2 , elle prend des vecteurs de \mathbb{R}^2 et les transforme en vecteurs de \mathbb{R}^3 , ajoutant ainsi une dimension.

Exemple :

Considérons la matrice suivante :

$$N = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

Cette matrice est de taille 3×2 et prend des vecteurs de \mathbb{R}^2 pour les transformer en vecteurs de \mathbb{R}^3 , en ajoutant une troisième composante nulle.

Application de la matrice :

Prenons un vecteur $w \in \mathbb{R}^2$:

$$w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

Alors,

$$Nw = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ 0 \end{pmatrix}$$

La transformation N prend le vecteur w et produit un vecteur de \mathbb{R}^3 en ajoutant une troisième composante nulle.

4.5 Addition et soustraction de matrices

L'addition ou la soustraction de deux matrices de même dimension se fait élément par élément. Si $A = [a_{ij}]$ et $B = [b_{ij}]$ sont deux matrices, alors $A + B = [a_{ij} + b_{ij}]$ et $A - B = [a_{ij} - b_{ij}]$.

Ainsi:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}.$$

$$A + B = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}.$$

```
import numpy as np

# Matrices A et B
A = np.array([[1, 2],
              [3, 4]])

B = np.array([[5, 6],
              [7, 8]])

# Addition
C = A + B
print("A + B =\n", C)

# Soustraction
D = A - B
print("A - B =\n", D)
```

4.6 Multiplication de 2 Matrices

La multiplication de deux matrices est possible lorsque le nombre de colonnes de la première matrice correspond au nombre de lignes de la deuxième.

Si A est une matrice de dimension $m \times n$ et B est une matrice de dimension $n \times p$, le produit AB est une matrice de dimension $m \times p$ où chaque élément c_{ij} est calculé comme le produit scalaire de la ligne i de A et de la colonne j de B , c'est-à-dire

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

En terme de dimensions, on a $(m, n) * (n, p) = (m, p)$.

Exemple

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \quad \text{et} \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}$$

Ici on peut confirmer que le produit matriciel $A * B$ est possible car la matrice A a autant de colonnes que la matrice B a de lignes. En effet la matrice A est de taille $(3, 2)$ et la matrice B est de taille $(2, 3)$, et on obtiendra une matrice de taille $(3, 3)$.

En terme de dimensions on a $(3, 2) * (2, 3) = (3, 3)$.

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

```

import numpy as np

# Matrice A de dimension (3, 2)
A = np.array([[1, 2],
              [3, 4],
              [5, 6]])

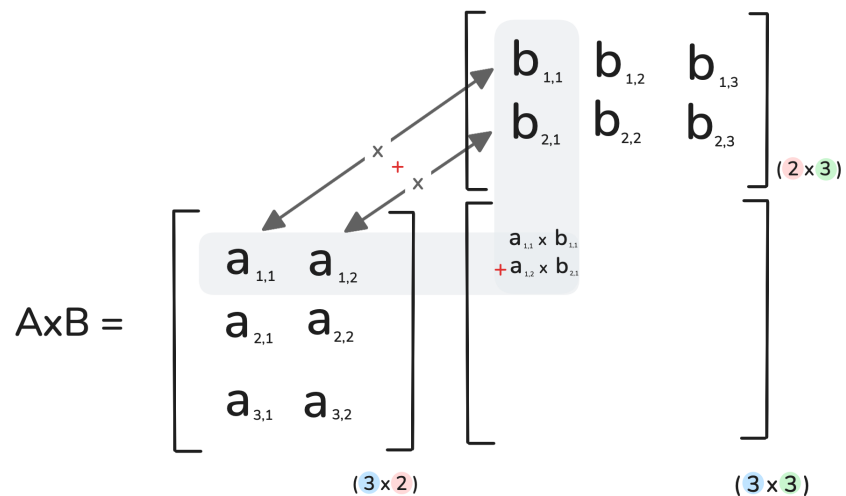
# Matrice B de dimension (2, 3)
B = np.array([[7, 8, 9],
              [10, 11, 12]])

# Multiplication de A par B
C = A @ B
print("A * B =\n", C)

# Dimensions de A, B, et C
print("Dimensions de A :", A.shape)
print("Dimensions de B :", B.shape)
print("Dimensions de C :", C.shape)

```

Pour plus de détails, voici une visualisation du calcul du premier élément de la matrice $A * B$. Pour plus de simplicité lorsque l'on pose le calcul, on place souvent la deuxième matrice en haut pour pouvoir mieux associer les lignes de la première matrice avec les colonnes de la deuxième.



Propriétés des opérations matricielles

1. $A + B = B + A$ (commutativité de l'addition)
2. $A + (B + C) = (A + B) + C$ (associativité de l'addition)
3. $(AB)C = A(BC)$ (associativité de la multiplication)
4. $A(B + C) = AB + AC$ (distributivité de la multiplication par rapport à l'addition)

Remarque importante

Contrairement à l'addition, **la multiplication de matrices n'est pas commutative en général**, c'est-à-dire que $AB \neq BA$ dans la plupart des cas. Par exemple, si nous prenons les matrices suivantes :

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{et} \quad B = \begin{bmatrix} 2 & 0 \\ 1 & 2 \end{bmatrix}$$

Calculons AB et BA :

$$AB = \begin{bmatrix} 1 \cdot 2 + 2 \cdot 1 & 1 \cdot 0 + 2 \cdot 2 \\ 3 \cdot 2 + 4 \cdot 1 & 3 \cdot 0 + 4 \cdot 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 10 & 8 \end{bmatrix}$$

$$BA = \begin{bmatrix} 2 \cdot 1 + 0 \cdot 3 & 2 \cdot 2 + 0 \cdot 4 \\ 1 \cdot 1 + 2 \cdot 3 & 1 \cdot 2 + 2 \cdot 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 7 & 10 \end{bmatrix}$$

On voit clairement que $AB \neq BA$.

4.7 Transposée d'une Matrice

La transposée d'une matrice $A = [a_{ij}]$ est la matrice A^T obtenue en échangeant les lignes et les colonnes de A . Ainsi, si A est de dimension $m \times n$, alors A^T sera de dimension $n \times m$, et $[A^T]_{ij} = a_{ji}$.

Définition formelle

On appelle transposée de $A \in \mathcal{M}_{q,p}(\mathbb{K})$ (matrice de dimension q,p à coefficients dans \mathbb{K}) la matrice notée $A^T \in \mathcal{M}_{p,q}(\mathbb{K})$ dont les colonnes sont formées par les lignes de A . Autrement dit :

$$\forall i \in \llbracket 1, p \rrbracket, \quad \forall j \in \llbracket 1, q \rrbracket, \quad [A^T]_{ij} = a_{ji}$$

Remarque Transposer revient à échanger les lignes et les colonnes d'une matrice.

Exemple Si $A = \begin{bmatrix} 1 & -3 \\ 2 & 7 \\ 0 & -4 \end{bmatrix}$, alors $A^T = \begin{bmatrix} 1 & 2 & 0 \\ -3 & 7 & -4 \end{bmatrix}$.

Propriétés de la transposée

- $(A^T)^T = A$
- $(A + B)^T = A^T + B^T$
- $(\lambda A)^T = \lambda A^T$ pour tout scalaire λ
- $(AB)^T = B^T A^T$ **Attention on change l'ordre**

Visuellement, voici l'impact de la transposée sur les lignes:

$$A = \begin{bmatrix} \begin{array}{c} a_{1,1} \\ a_{2,1} \\ \vdots \\ a_{n,1} \end{array} & \begin{array}{c} a_{1,2} \\ a_{2,2} \\ \vdots \\ a_{n,2} \end{array} & \dots & \begin{array}{c} a_{1,n} \\ a_{2,n} \\ \vdots \\ a_{n,m} \end{array} \end{bmatrix} \quad A^T = \begin{bmatrix} a_{1,1} & a_{2,1} & \dots & a_{n,1} \\ a_{1,2} & a_{2,2} & \dots & a_{n,2} \\ \vdots & \vdots & & \vdots \\ a_{1,n} & a_{2,n} & \dots & a_{n,m} \end{bmatrix}$$

Remarque : Si on multiplie une matrice rectangle par sa transposée, alors le résultat est toujours une matrice **symétrique** !

```
import numpy as np

# Matrice A
A = np.array([[1, -3],
              [2, 7],
              [0, -4]])

# Transposée de A
A_T = A.T
print("A :\n", A)
print("Transposée de A :\n", A_T)
```

4.8 Trace d'une Matrice

La trace d'une matrice A est la somme de ses éléments diagonaux, c'est-à-dire les éléments a_{ii} . Elle est notée $\text{Tr}(A)$.

Définition

Soit $A = (a_{ij}) \in \mathcal{M}_n(\mathbb{K})$ une matrice carrée. On appelle **trace** de A et on note $\text{Tr}(A)$, le scalaire :

$$\text{Tr}(A) = \sum_{i=1}^n a_{i,i}$$

Exemple

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

La trace de A est :

$$\text{Tr}(A) = a_{11} + a_{22} + a_{33}$$

```
import numpy as np

# Matrice carrée A
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# Calcul de la trace de A
trace = np.trace(A)
print("Trace de A :", trace)
```

Propriétés de la trace

- La trace est une forme linéaire. En particulier, si $\alpha, \beta \in \mathbb{K}$ et si $A, B \in \mathcal{M}_n(\mathbb{K})$, alors

$$\text{Tr}(\alpha A + \beta B) = \alpha \text{Tr}(A) + \beta \text{Tr}(B)$$

- Pour toutes matrices $A, B \in \mathcal{M}_n(\mathbb{K})$,

$$\text{Tr}(AB) = \text{Tr}(BA)$$

4.9 Rang d'une Matrice

Le rang d'une matrice A , noté $\text{rang}(A)$, est le nombre maximal de colonnes de A qui sont linéairement indépendantes.

Définition

On appelle rang de $A \in \mathcal{M}_{q,p}(\mathbb{K})$, et on note $\text{rg}(A)$, le rang de la famille constituée des vecteurs colonnes $\vec{C}_1, \dots, \vec{C}_p$ de A dans \mathbb{K}^q .

Propriétés du rang

- Le rang d'une matrice ayant les colonnes $\vec{C}_1, \vec{C}_2, \dots, \vec{C}_p$ n'est pas modifié par les trois opérations élémentaires suivantes sur les vecteurs :
 1. $\vec{C}_i \leftarrow \lambda \vec{C}_i$ avec $\lambda \neq 0$: on peut multiplier une colonne par un scalaire non nul.
 2. $\vec{C}_i \leftarrow \vec{C}_i + \lambda \vec{C}_j$ avec $\lambda \in \mathbb{K}$ (et $j \neq i$) : on peut ajouter à la colonne \vec{C}_i un multiple d'une autre colonne \vec{C}_j .
 3. $\vec{C}_i \leftrightarrow \vec{C}_j$: on peut échanger deux colonnes.
- Le rang d'une matrice est égal au rang de sa transposée :

$$\text{rg}(A) = \text{rg}(A^T)$$

Lien avec les Variables d'un Dataset Dans le contexte du machine learning, considérons X , une matrice où chaque colonne représente une variable différente d'un ensemble de données et chaque ligne une observation. Le rang de X nous informe sur la redondance entre les variables :

- Un rang plein (égal au nombre de colonnes de X) indique que toutes les variables sont linéairement indépendantes. Aucune variable n'est une combinaison linéaire des autres, ce qui signifie qu'elles apportent toutes des informations uniques, sans redondance.
- Un rang inférieur au nombre de colonnes suggère que certaines variables sont redondantes. Cela peut indiquer la présence de multicollinéarité, où une ou plusieurs variables sont des combinaisons linéaires d'autres. Cela est crucial en régression linéaire, car la multicollinéarité peut affecter la stabilité des estimations des coefficients.

Exemple Considérons la matrice A suivante :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Les colonnes de A sont linéairement dépendantes, car la troisième colonne peut être exprimée comme une combinaison linéaire des deux premières :

$$\begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix} = -1 \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix} + 2 \begin{bmatrix} 2 \\ 5 \\ 8 \end{bmatrix}$$

Le rang de A est donc 2, car il y a 2 colonnes linéairement indépendantes.

```
import numpy as np

# Matrice A
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# Calcul du rang de A
rang = np.linalg.matrix_rank(A)
print("Rang de A :", rang)
```

Remarque : Comme on sait que le rang d'une matrice est le même que le rang de sa transposée, on peut également calculer le rang suivant les lignes.

4.10 Matrice orthogonale

Une matrice orthogonale est une matrice réelle $A \in \mathcal{M}_n(\mathbb{R})$ qui vérifie :

$$A^T A = A A^T = I_n$$

5 Espaces vectoriels

5.1 Définition et exemple

Soit \mathbb{K} un corps. On appelle **espace vectoriel** sur le corps \mathbb{K} tout ensemble E non vide muni d'une loi de composition interne (addition) $+$: $E \times E \rightarrow E$ et d'une loi de composition externe (multiplication par un scalaire) \cdot : $\mathbb{K} \times E \rightarrow E$ telles que :

1. $(E, +)$ est un groupe commutatif. On note 0_E son élément neutre.
2. Pour tout $\alpha, \beta \in \mathbb{K}$ et pour tout $x, y \in E$, on a :
 - (a) $\alpha \cdot (x + y) = \alpha \cdot x + \alpha \cdot y$
 - (b) $(\alpha + \beta) \cdot x = \alpha \cdot x + \beta \cdot x$
 - (c) $\alpha \cdot (\beta \cdot x) = (\alpha\beta) \cdot x$
 - (d) $1_{\mathbb{K}} \cdot x = x$

Les éléments de \mathbb{K} sont appelés scalaires, ceux de E sont appelés vecteurs. L'élément neutre de $(E, +)$, 0_E , est appelé **vecteur nul**.

Un espace vectoriel est un ensemble sur lequel on peut définir une addition et une multiplication par un scalaire et rien de plus, si ce n'est que cette addition et cette multiplication doivent vérifier les axiomes ci-dessus. Vous connaissez un certain nombre d'ensembles de ce type :

- \mathbb{R} (ou \mathbb{C}) est un espace vectoriel.
- L'ensemble des vecteurs du plan \mathbb{R}^2 ou de l'espace \mathbb{R}^3 est un espace vectoriel.
- L'ensemble $\mathcal{F}(I, \mathbb{R})$ des fonctions de I dans \mathbb{R} (ou \mathbb{C}) est un espace vectoriel.
- Les ensembles de polynômes à coefficients réels et ou complexes sont des espaces vectoriels.

5.2 Sous-espace vectoriels (SEV)

5.2.1 Définition

Un sous-espace vectoriel F d'un espace vectoriel E est un sous-ensemble F de E tel que :

- (SEV1) $0_E \in F$;
- (SEV2) $\forall x, y \in F$, on a $x + y \in F$;
- (SEV3) $\forall \lambda \in \mathbb{K}$ et $\forall x \in F$, on a $\lambda \cdot x \in F$.

5.2.2 Lemme

Si $F \subseteq E$ vérifie les points (SEV2) et (SEV3) et est non vide, alors c'est un sous-espace vectoriel. Autrement dit, le point (SEV1) est automatiquement vérifié.

5.2.3 Théorème: Dimension d'un SEV

Soit E un espace vectoriel de dimension finie n et F un sous-espace vectoriel de E .

1. F est de dimension finie et $\dim F \leq \dim E$;
2. $\dim F = \dim E \iff F = E$.

Remarque On utilise souvent ce résultat pour montrer que deux sous-espaces vectoriels F et G sont égaux :

$$F \subseteq G \text{ et } \dim F = \dim G \implies F = G$$

5.3 Somme directe de sous-espaces vectoriels

5.3.1 Définition

Soient E un espace vectoriel sur \mathbb{K} et F et G deux sous-espaces vectoriels de E . On dit que E est la **somme directe** de F et G , notée $E = F \oplus G$, si :

- $E = F + G$ (tout vecteur de E s'écrit comme somme d'un vecteur de F et d'un vecteur de G) ;
- $F \cap G = \{0_E\}$ (l'intersection de F et G est réduite au vecteur nul).

Propriété : La somme directe de deux sous-espaces vectoriels correspond à la réunion de deux espaces sans recouvrement, permettant une décomposition unique des vecteurs de E .

5.3.2 Généralisation

Cette notion peut être étendue à une famille finie de sous-espaces vectoriels F_1, F_2, \dots, F_k .

E est la **somme directe** des F_i , notée $E = \bigoplus_{i=1}^k F_i$, si :

- $E = F_1 + F_2 + \dots + F_k$ (tout vecteur de E s'écrit comme somme de vecteurs des F_i) ;
- Pour toute famille $(x_i)_{i=1}^k$ avec $x_i \in F_i$, si $x_1 + x_2 + \dots + x_k = 0_E$, alors $x_i = 0_E$ pour tout i .

Exemple: Considérons l'espace vectoriel $E = \mathbb{R}^2$, le plan euclidien.

- Soit F l'axe des abscisses, constitué de tous les vecteurs de la forme $(x, 0)$
- Soit G l'axe des ordonnées, constitué de tous les vecteurs de la forme $(0, y)$

Vérifions les deux conditions :

- $E = F + G$: Tout vecteur $(x, y) \in \mathbb{R}^2$ peut s'écrire comme $(x, 0) + (0, y)$, avec $(x, 0) \in F$ et $(0, y) \in G$
- $F \cap G = \{(0, 0)\}$: Les seuls vecteurs communs à F et G sont ceux pour lesquels $x = 0$ et $y = 0$, donc le vecteur nul

Ainsi, nous pouvons dire que $\mathbb{R}^2 = F \oplus G$.

5.4 Notions affines et vectorielles

Il est essentiel de bien distinguer les notions vectorielles des notions affines. Les espaces affines sont ceux que vous connaissez depuis toujours :

- Un vecteur est donné par deux points, un point de départ A et un point d'arrivée B . On dit que deux vecteurs \vec{AB} et \vec{CD} sont égaux si $ABDC$ est un parallélogramme.
- Une droite est composée de points. Elle est donnée par un point par lequel elle passe et par un vecteur qui la dirige. On parle de droite affine.
- Un plan est lui aussi composé de points. Il est donné par un point par lequel il passe et par deux vecteurs qui l'engendrent. On parle de plan affine.

Si on fixe une origine O dans notre espace affine, ses points peuvent être identifiés aux vecteurs d'origine O . Ce sont ces vecteurs qui nous intéressent dans un espace vectoriel. La notion de point n'a pas de sens. Dans un espace vectoriel :

- Tous les éléments sont des vecteurs. On peut les représenter comme les vecteurs d'origine O d'un espace affine.
- Une droite est engendrée par un vecteur u (non nul) et est formée de vecteurs. On peut se l'imaginer dans un espace affine comme une droite passant par O et dirigée par u .
- Un plan est engendré par deux vecteurs u et v (non colinéaires) et est formé de vecteurs. On peut se l'imaginer dans un espace affine comme un plan passant par O et engendré par u et v .

Dans toute la suite, \mathbb{K} est un corps (\mathbb{R} ou \mathbb{C}). \mathbf{E} et \mathbf{F} désignent des \mathbb{K} -espaces vectoriels.

5.5 Combinaisons linéaires

Soit $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n\}$ une famille de n vecteurs. On appelle **combinaison linéaire** de ces n vecteurs tout vecteur $\vec{v} \in E$ de la forme :

$$\vec{v} = \lambda_1 \vec{v}_1 + \lambda_2 \vec{v}_2 + \dots + \lambda_n \vec{v}_n = \sum_{k=1}^n \lambda_k \vec{v}_k$$

où $\lambda_1, \lambda_2, \dots, \lambda_n \in \mathbb{K}$.

Exemple : Dans \mathbb{R}^2 , soient $v_1 = (1, 0)$ et $v_2 = (0, 1)$. Tout vecteur $v = (x, y)$ peut s'exprimer comme une combinaison linéaire de v_1 et v_2 :

$$v = xv_1 + yv_2.$$

5.6 Familles libres (linéairement indépendantes)

On dit qu'une famille $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_p\}$ de vecteurs est **liée**, ou que les vecteurs $\vec{v}_1, \dots, \vec{v}_p$ sont **linéairement dépendants** si et seulement si un des vecteurs de la famille est combinaison linéaire des autres ou autrement dit s'il existe un p -uplet $(\lambda_1, \dots, \lambda_p) \in \mathbb{K}^p$ de scalaires non tous nuls vérifiant :

$$\lambda_1 \vec{v}_1 + \lambda_2 \vec{v}_2 + \dots + \lambda_p \vec{v}_p = \vec{0}$$

Sinon, on dit que la famille $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_p\}$ est **libre**, ou que les vecteurs $\vec{v}_1, \dots, \vec{v}_p$ sont **linéairement indépendants**. Cela signifie que la seule solution de l'équation suivante est celle où tous les coefficients sont nuls :

$$\lambda_1 \vec{v}_1 + \lambda_2 \vec{v}_2 + \dots + \lambda_p \vec{v}_p = \vec{0} \implies \lambda_1 = \lambda_2 = \dots = \lambda_p = 0$$

Remarques

- Une famille contenant le vecteur nul n'est jamais libre.
- La famille vide est libre.
- Une famille extraite d'une famille libre est libre.

Exemple

- Trois vecteurs du plan sont toujours liés. De même, 4 vecteurs de l'espace sont toujours liés.
- Dans \mathbb{R}^4 , les vecteurs $\vec{u}_1 = (1, 0, 1, -1)$, $\vec{u}_2 = (3, -2, 2, -3)$ et $\vec{u}_3 = (-1, 2, 0, 1)$ forment une famille liée. En effet, $\vec{u}_2 = 2\vec{u}_1 - \vec{u}_3$.
- Dans $\mathcal{F}(\mathbb{R}, \mathbb{R})$ (les fonctions de \mathbb{R} dans \mathbb{R}), les fonctions $f_1 : x \mapsto \cos^2 x$, $f_2 : x \mapsto \sin^2 x$ et $f_3 : x \mapsto \cos 2x$ sont liées. En effet, $f_3 = f_1 - f_2$.

5.7 Familles génératrices

On dit qu'une famille $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_p\}$ de p vecteurs de E **engendre** l'espace vectoriel E (ou est **génératrice** de E) si tout vecteur de E peut s'exprimer comme combinaison linéaire de la famille $\{\vec{v}_1, \dots, \vec{v}_p\}$:

$$\forall \vec{v} \in E, \exists (\lambda_1, \dots, \lambda_p) \in \mathbb{K}^p \text{ tel que } \vec{v} = \sum_{k=1}^p \lambda_k \vec{v}_k$$

Autrement dit : $\text{Vect}(\{\vec{v}_1, \dots, \vec{v}_p\}) = E$.

Exemple

- Dans \mathbb{R}^2 , soient $\vec{x}_1 = (1, 0)$ et $\vec{x}_2 = (0, 1)$. La famille (\vec{x}_1, \vec{x}_2) est génératrice de \mathbb{R}^2 . En effet, si $\vec{v} = (x, y) \in \mathbb{R}^2$, alors $\vec{v} = x\vec{x}_1 + y\vec{x}_2$.
- On montre de même que la famille $((1, 0, 0), (0, 1, 0), (0, 0, 1))$ engendre \mathbb{R}^3 .

5.8 Bases et dimension

5.8.1 Définition d'une base

On dit qu'une famille $\{\vec{v}_1, \dots, \vec{v}_p\}$ de vecteurs de E est une **base** de E si et seulement si, à la fois :

1. $\{\vec{v}_1, \dots, \vec{v}_p\}$ est une famille libre de E .
2. $\{\vec{v}_1, \dots, \vec{v}_p\}$ est une famille génératrice de E .

Exemples

- La famille $(1, i)$ est une base du \mathbb{R} -espace vectoriel \mathbb{C} . En effet, si $a, b \in \mathbb{R}$ sont tels que $a \cdot 1 + b \cdot i = 0$, alors $a + ib = 0$ et donc $a = b = 0$. La famille est donc libre. Pour tout nombre complexe z , il existe $a, b \in \mathbb{R}$ tels que $z = a + ib$. La famille est donc aussi génératrice de \mathbb{C} . C'est donc une base de \mathbb{C} .
- Dans le plan, deux vecteurs non colinéaires forment une base du plan.
- Dans \mathbb{R}^3 , la famille $\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ forme une base de \mathbb{R}^3 .

5.8.2 Base canonique de \mathbb{K}^n

Soit $n \in \mathbb{N}^*$. Considérons le \mathbb{K} -espace vectoriel $E = \mathbb{K}^n$. Il existe une base privilégiée $\{\vec{e}_1, \dots, \vec{e}_n\}$ de \mathbb{K}^n dite **canonique** et donnée par :

$$\vec{e}_1 = (1, 0, \dots, 0), \quad \vec{e}_2 = (0, 1, 0, \dots, 0), \quad \dots, \quad \vec{e}_n = (0, 0, \dots, 1)$$

5.8.3 Composantes d'un vecteur relativement à une base

Une famille $\vec{e} = \{\vec{e}_1, \dots, \vec{e}_n\}$ de E est une base de E si et seulement si, pour tout vecteur $\vec{v} \in E$, il existe une unique famille de scalaires $(\lambda_1, \dots, \lambda_n) \in \mathbb{K}^n$ telle que :

$$\vec{v} = \sum_{k=1}^n \lambda_k \vec{e}_k$$

Le n -uplet $(\lambda_1, \dots, \lambda_n)$ est alors appelé famille des **composantes** (ou **coordonnées**) de \vec{v} dans la base \vec{e} .

5.8.4 Espace vectoriel de dimension finie

On dit qu'un \mathbb{K} -espace vectoriel est de **dimension finie** s'il admet une famille génératrice finie. Dans le cas contraire, on dit que E est de **dimension infinie**. De plus, par convention, on dit que $E = \{0\}$ est un espace de dimension finie.

Exemple

- \mathbb{K}^n est de dimension finie car sa base canonique est une famille génératrice finie de \mathbb{K}^n .

5.8.5 Dimension d'un espace vectoriel

- Si $E = \{0\}$, on dit que E est de **dimension 0** et on note $\dim E = 0$.
- Sinon, si E est un espace vectoriel de dimension finie non réduit à $\{0\}$, on appelle **dimension** de E le cardinal d'une base de E (le nombre de vecteurs dans la base) et on le note $\dim E$.

Exemple

- $\dim \mathbb{K}^n = n$.

6 Applications Linéaires

6.1 Définition

Soient E et F deux \mathbb{R} -espaces vectoriels. Une application $u : E \rightarrow F$ est dite linéaire si et seulement si, pour tous $x, y \in E$ et $\lambda, \mu \in \mathbb{R}$, on a :

$$u(\lambda x + \mu y) = \lambda u(x) + \mu u(y).$$

Lorsque $E = F$, on appelle une telle application un **endomorphisme**.

6.2 Exemples

1. **Application Nulle** : Pour deux espaces vectoriels E et F , l'application nulle, qui associe à tout $x \in E$ le vecteur nul 0_F , est linéaire.
2. **Multiplication par un scalaire** : L'application $x \mapsto 2x$ de \mathbb{R} dans \mathbb{R} est linéaire. En revanche, l'application $x \mapsto x^2$ ne l'est pas.
3. **Évaluation en un Point** : Pour $x_0 \in \mathbb{R}$, l'application d'évaluation $\text{ev}_{x_0} : F(\mathbb{R}, \mathbb{R}) \rightarrow \mathbb{R}$ qui associe à une fonction f sa valeur en x_0 , $f(x_0)$, est linéaire.
4. **Dérivation** : L'application de dérivation $d : \mathbb{R}[X] \rightarrow \mathbb{R}[X]$, qui associe à un polynôme P son dérivé P' , est un endomorphisme de $\mathbb{R}[X]$.

6.3 Propriétés

L'ensemble des applications linéaires de E dans F est lui-même un \mathbb{R} -espace vectoriel. Cela signifie que, pour deux applications linéaires u et v de E dans F , et pour tous scalaires $\alpha, \beta \in \mathbb{R}$, l'application $\alpha u + \beta v$ est également linéaire. Cet espace vectoriel est noté $L(E, F)$.

6.4 Isomorphisme

Soient E et F deux \mathbb{R} -espaces vectoriels. Une application linéaire $u : E \rightarrow F$ est dite **isomorphisme** si elle est **bijective**, c'est-à-dire qu'elle est à la fois **injective** et **surjective**.

Définition formelle :

- **Injective** : Une application u est injective si, pour tous $x_1, x_2 \in E$, $u(x_1) = u(x_2)$ implique $x_1 = x_2$. En d'autres termes, chaque élément de E est envoyé sur un élément distinct de F .
- **Surjective** : Une application u est surjective si, pour tout $y \in F$, il existe au moins un $x \in E$ tel que $u(x) = y$. Cela signifie que l'application couvre entièrement l'espace F .

Ainsi, un isomorphisme établit une correspondance un-à-un et surjective entre les éléments de E et ceux de F , garantissant que les deux espaces vectoriels sont structurellement identiques en termes d'algèbre linéaire.

Notion de bijectivité :

La bijectivité est une propriété cruciale dans la définition des isomorphismes. Une application bijective possède deux caractéristiques principales :

1. **Injectivité** : Aucun élément distinct de l'espace de départ E n'est envoyé sur le même élément de l'espace d'arrivée F . Cela garantit qu'il n'y a pas de "doublons" dans l'image de l'application.

2. **Surjectivité** : Chaque élément de l'espace d'arrivée F est l'image d'au moins un élément de l'espace de départ E . Cela assure que l'application couvre tout l'espace cible sans laisser de "trous".

En combinant ces deux propriétés, une application bijective permet de revenir en arrière de manière unique, c'est-à-dire qu'il existe une application inverse $u^{-1} : F \rightarrow E$ telle que :

$$u^{-1}(u(x)) = x \quad \text{pour tout } x \in E \quad \text{et} \quad u(u^{-1}(y)) = y \quad \text{pour tout } y \in F.$$

Cela signifie que les espaces E et F sont parfaitement correspondants l'un à l'autre par l'intermédiaire de l'isomorphisme u .

6.5 Noyau et Image

Pour une application linéaire $u : E \rightarrow F$:

- **Noyau** : Le noyau de u , noté $\ker(u)$, est l'ensemble des vecteurs de E envoyés sur le vecteur nul de F :

$$\ker(u) = \{x \in E \mid u(x) = 0_F\}.$$

- **Image** : L'image de u , notée $\text{Im}(u)$, est l'ensemble des vecteurs de F qui sont des images de vecteurs de E :

$$\text{Im}(u) = \{u(x) \mid x \in E\}.$$

6.6 Matrice d'une Application Linéaire

Toute application linéaire entre espaces vectoriels peut être représentée par une matrice. Si $B = (e_1, \dots, e_n)$ est une base de E et $C = (f_1, \dots, f_m)$ une base de F , alors pour chaque j , $u(e_j)$ peut être écrit comme une combinaison linéaire des f_i :

$$u(e_j) = \sum_{i=1}^m a_{ij} f_i.$$

La matrice de u dans les bases B et C est la matrice $m \times n$ dont les coefficients sont a_{ij} .

6.6.1 Matrices Diagonales et Étirement des Axes

Les matrices diagonales sont des matrices carrées où tous les éléments hors diagonale sont nuls. Elles jouent un rôle essentiel dans les transformations linéaires car elles effectuent des **étirements** ou des **contractions** le long des axes principaux.

Par exemple, considérons la matrice diagonale suivante :

$$D = \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix}$$

Cette matrice étire ou contracte le vecteur d'entrée le long des axes x et y par des facteurs k_1 et k_2 , respectivement.

Exemple Numérique :

Prenons :

$$D = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$$

Appliquons cette matrice aux vecteurs de la base canonique $e_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ et $e_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$:

$$De_1 = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

$$De_2 = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$$

Ainsi, le vecteur e_1 est **étiré** par un facteur 2 le long de l'axe x , et le vecteur e_2 est **étiré** par un facteur 3 le long de l'axe y .

6.6.2 Matrices de Rotation

Les matrices de rotation sont des matrices orthogonales qui effectuent une rotation des vecteurs dans le plan. En deux dimensions, une matrice de rotation d'un angle θ est donnée par :

$$R(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

Cette matrice effectue une rotation antihoraire d'un angle θ autour de l'origine.

Exemple Numérique :

Considérons une rotation de 90 degrés ($\theta = 90^\circ$ ou $\theta = \frac{\pi}{2}$ radians) :

$$R\left(\frac{\pi}{2}\right) = \begin{pmatrix} \cos \frac{\pi}{2} & -\sin \frac{\pi}{2} \\ \sin \frac{\pi}{2} & \cos \frac{\pi}{2} \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

Appliquons cette matrice aux vecteurs de la base canonique $e_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ et $e_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$:

$$R\left(\frac{\pi}{2}\right) e_1 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$R\left(\frac{\pi}{2}\right) e_2 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

Ainsi, le vecteur e_1 est transformé en e_2 , et le vecteur e_2 est transformé en $-e_1$, ce qui correspond à une rotation de 90 degrés dans le sens antihoraire.

6.7 Matrices de Passage

Pour deux bases B et B' d'un même espace vectoriel E , la matrice de passage de B à B' est la matrice dont les colonnes sont les coordonnées des vecteurs de B' dans la base B .

6.8 Composition et Inverse

- **Composition** : La matrice de la composition de deux applications linéaires est le produit des matrices de ces applications :

$$\text{Mat}(v \circ u) = \text{Mat}(v) \cdot \text{Mat}(u).$$

- **Inverse** : Si u est un isomorphisme, la matrice de son inverse est l'inverse de la matrice de u :

$$\text{Mat}(u^{-1}) = (\text{Mat}(u))^{-1}.$$

Remarque : Dans ce cours, nous parlerons indifféremment d'application linéaire et de matrice d'une application linéaire, car toute application linéaire peut être représentée par une matrice, et les propriétés des applications linéaires peuvent souvent être étudiées à travers leurs matrices associées.

6.9 Exemple d'Application en Machine Learning

En machine learning, les transformations linéaires peuvent être utilisées pour normaliser les données, réduire la dimensionnalité, ou encore projeter les données dans un espace où elles sont plus facilement séparables. Les coefficients de la matrice représentant une transformation linéaire correspondent alors aux poids appris par un modèle de régression linéaire ou par une couche linéaire d'un réseau de neurones. Nous verrons tout ça par la suite.

7 Déterminant d'une Matrice

Le déterminant d'une matrice est une valeur numérique associée à une matrice carrée (et seulement carrée). Il est utilisé pour diverses applications en algèbre linéaire, telles que la résolution de systèmes d'équations linéaires, la diagonalisation des matrices et la détermination de l'aire ou du volume dans la géométrie.

7.1 Définition

Soit A une matrice carrée d'ordre n . Le déterminant de A , noté $\det(A)$ ou $|A|$, est une fonction qui associe à A un scalaire, calculé de manière récursive selon la taille de la matrice.

7.2 Calcul du Déterminant

Le calcul du déterminant d'une matrice dépend de sa taille et de sa structure. Voici les méthodes courantes pour le calculer :

7.2.1 Matrice 2×2

Pour une matrice 2×2 :

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Le déterminant est calculé comme suit :

$$\det(A) = ad - bc.$$

7.2.2 Matrice 3×3

Pour une matrice 3×3 :

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

Le déterminant peut être calculé à la main en utilisant la règle de Sarrus (uniquement pour les matrices 3×3) ou le développement par cofacteurs (pour n'importe quelle taille de matrice). Dans Numpy, pour calculer le déterminant, la matrice est décomposée suivant la décomposition LU.

Règle de Sarrus La règle de Sarrus consiste à étendre la matrice en recopiant les deux premières colonnes à droite et en additionnant les diagonales (avec un moins si la diagonale monte, en lisant de gauche à droite) :

$$\det(A) = aei + bfg + cdh - (gec + hfa + idb)$$

$$\det(A) = \begin{vmatrix} a & b & c & a & b \\ d & e & f & d & e \\ g & h & i & g & h \end{vmatrix}$$

$$= +aei + bfg + cdh - gec - hfa - idb$$

```
import numpy as np

# Calcul du déterminant en utilisant la règle de Sarrus
# pour une matrice 3x3
def det_sarrus(A):
    if A.shape != (3, 3):
        raise ValueError("La matrice doit être de taille
3x3.")
    a, b, c = A[0]
    d, e, f = A[1]
    g, h, i = A[2]
    return a*e*i + b*f*g + c*d*h - c*e*g - b*d*i - a*f*h

# Exemple de matrice 3x3
A = np.array([[2, 1, 3],
              [0, -1, 4],
              [5, 2, 0]])

det_A = det_sarrus(A)
print("Le déterminant de A (règle de Sarrus) est :",
      det_A)
```


7.2.3 Développement par Cofacteurs

Le développement par cofacteurs (ou par mineurs) est une méthode générale pour calculer le déterminant d'une matrice de taille $n \times n$.

Mineur et Cofacteur

- **Mineur** M_{ij} : Le déterminant de la matrice obtenue en supprimant la i -ème ligne et la j -ème colonne de A .
- **Cofacteur** C_{ij} : $C_{ij} = (-1)^{i+j} M_{ij}$.

Formule du Développement Le déterminant de A peut être calculé en développant par rapport à une ligne (ou une colonne) :

$$\det(A) = \sum_{j=1}^n a_{ij} C_{ij} \quad (\text{développement selon la ligne } i)$$

ou

$$\det(A) = \sum_{i=1}^n a_{ij} C_{ij} \quad (\text{développement selon la colonne } j)$$

Exemple Calculons le déterminant de la matrice 3×3 suivante en développant par rapport à la première ligne :

$$A = \begin{bmatrix} 2 & 1 & 3 \\ 0 & -1 & 4 \\ 5 & 2 & 0 \end{bmatrix}$$

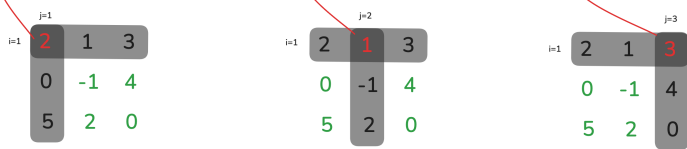
Les cofacteurs sont :

- $C_{11} = (-1)^{1+1} \begin{vmatrix} -1 & 4 \\ 2 & 0 \end{vmatrix} = 1 * (-1 * 0 - 4 * 2) = -8$
- $C_{12} = (-1)^{1+2} \begin{vmatrix} 0 & 4 \\ 5 & 0 \end{vmatrix} = (-1)(0 * 0 - 4 * 5) = 20$
- $C_{13} = (-1)^{1+3} \begin{vmatrix} 0 & -1 \\ 5 & 2 \end{vmatrix} = 1 * (0 * 2 - (-1) * 5) = 5$

Le déterminant est :

$$\det(A) = a_{11}C_{11} + a_{12}C_{12} + a_{13}C_{13} = 2 * (-8) + 20 + 3 * 5 = -16 + 20 + 15 = 19$$

$$\det(A) = \begin{vmatrix} 2 & 1 & 3 \\ 0 & -1 & 4 \\ 5 & 2 & 0 \end{vmatrix}$$

$$= (-1)^{1+1} \times 2 \times \begin{vmatrix} -1 & 4 \\ 2 & 0 \end{vmatrix} + (-1)^{1+2} \times 1 \times \begin{vmatrix} 0 & 4 \\ 5 & 0 \end{vmatrix} + (-1)^{1+3} \times 3 \times \begin{vmatrix} 0 & -1 \\ 5 & 2 \end{vmatrix}$$


$$= (-1)^2 \times 2 \times -8 + (-1)^3 \times 1 \times -20 + (-1)^4 \times 3 \times 5$$

$$= -16 + 20 + 15$$

$$= 19$$

Ici, on a développé suivant la première ligne pour l'exemple, mais le but est de développer suivant une ligne ou une colonne qui contient le plus de 0, car cela fait des calculs en moins. Ici on aurait pu développer suivant la première colonne ou encore la 3^e ligne. On va également voir comment "créer" ces 0 dans la matrice pour calculer le déterminant plus facilement, dans la parties suivante. On peut également remarquer que si une colonne ou ligne est une combinaison linéaire des autres colonnes ou lignes (c'est à dire s'il y a une redondance de données) alors on a un déterminant nul, car on pourrait avoir une ligne ou colonne pleine de 0.

```
import numpy as np

# Calcul du déterminant d'une matrice 2x2
A = np.array([[2, 1, 3],
              [0, -1, 4],
              [5, 2, 0]])

det_A = np.linalg.det(A)
print("Le déterminant de A est :", det_A)
```

7.3 Propriétés des Opérations sur les Lignes et Colonnes

Le calcul du déterminant peut être simplifié en utilisant des opérations sur les lignes et les colonnes, en respectant les propriétés suivantes :

- **Échange de deux lignes (ou colonnes)** : Change le signe du déterminant.
- **Multiplication d'une ligne (ou colonne) par un scalaire k** : Multiplie le déterminant par k .
- **Ajout d'une combinaison linéaire d'une ligne (ou colonne) à une autre ligne (ou colonne)** : Ne change pas le déterminant.

Considérons la matrice suivante :

$$A = \begin{bmatrix} 2 & 1 & 3 \\ 0 & -1 & 4 \\ 5 & 2 & 0 \end{bmatrix}$$

Nous allons calculer le déterminant de A en simplifiant les calculs grâce aux opérations sur les lignes.

Étape 1 : Notre objectif est de transformer la matrice pour obtenir des zéros qui faciliteront le calcul du déterminant. Nous allons utiliser des opérations sur les lignes qui ne modifient pas la valeur du déterminant.

Étape 2 : Ajoutons à la ligne 3 une combinaison linéaire de la ligne 1 pour obtenir un zéro en position $(3, 1)$. • **Opération :** $L_3 \leftarrow L_3 - \frac{5}{2}L_1$

Calculons les nouvelles valeurs de L_3 :

- Position $(3, 1)$: $5 - \frac{5}{2} \times 2 = 5 - 5 = 0$
- Position $(3, 2)$: $-\frac{5}{2} \times 1 = 2 - \frac{5}{2} = -\frac{1}{2}$
- Position $(3, 3)$: $0 - \frac{5}{2} \times 3 = 0 - \frac{15}{2} = -\frac{15}{2}$

La matrice devient :

$$A' = \begin{bmatrix} 2 & 1 & 3 \\ 0 & -1 & 4 \\ 0 & -\frac{1}{2} & -\frac{15}{2} \end{bmatrix}$$

Remarque : L'ajout d'une combinaison linéaire d'une ligne à une autre ne change pas le déterminant.

Étape 3 : Avec ce zéro obtenu en position $(3, 1)$, nous pouvons plus facilement calculer le déterminant en développant selon la première colonne.

Étape 4 : Calcul du déterminant en développant selon la première colonne :

$$\det(A') = a_{11}C_{11} + a_{21}C_{21} + a_{31}C_{31}$$

Mais $a_{21} = 0$ et $a_{31} = 0$, donc :

$$\det(A') = a_{11}C_{11}$$

Calcul du cofacteur C_{11} :

- **Mineur M_{11} :** Déterminant de la sous-matrice obtenue en supprimant la première ligne et la première colonne :

$$M_{11} = \begin{vmatrix} -1 & 4 \\ -\frac{1}{2} & -\frac{15}{2} \end{vmatrix}$$

Calculons ce déterminant :

$$M_{11} = (-1) \times \left(-\frac{15}{2}\right) - 4 \times \left(-\frac{1}{2}\right) = \frac{15}{2} + 2 = \frac{19}{2}$$

- **Cofacteur C_{11} :**

$$C_{11} = (-1)^{1+1} M_{11} = (+1) \times \frac{19}{2} = \frac{19}{2}$$

Calcul du déterminant :

$$\det(A') = a_{11} C_{11} = 2 \times \frac{19}{2} = 19$$

Conclusion : Le déterminant de la matrice initiale A est $\det(A) = 19$. Comme cela était le cas dans l'exemple précédent.

7.4 Matrices Triangulaires et Diagonales

Pour une matrice **triangulaire** (supérieure ou inférieure) ou **diagonale**, le déterminant est le produit des éléments de la diagonale principale :

$$\det(A) = a_{11} \times a_{22} \times \cdots \times a_{nn}$$

Exemple Soit une matrice diagonale :

$$A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

Alors, $\det(A) = 2 \times 3 \times 4 = 24$.

Et soit une matrice triangulaire (ici triangulaire supérieure) :

$$A = \begin{bmatrix} 1 & 3 & 1 \\ 0 & 9 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

Alors, $\det(A) = 1 \times 9 \times 4 = 36$.

7.5 Propriétés du Déterminant

Le déterminant d'une matrice possède plusieurs propriétés importantes :

- **Déterminant de la Matrice Nulle** : $\det(\mathbf{0}_n) = 0$.
- **Déterminant de la Matrice Identité** : $\det(I_n) = 1$.
- **Homogénéité** : $\det(\lambda A) = \lambda^n \det(A)$ où λ est un scalaire.
- **Multiplicativité** : $\det(AB) = \det(A) \times \det(B)$.
- **Transposée** : $\det(A^\top) = \det(A)$.
- **Opérations sur les Lignes/Colonnes** :
 - Si deux lignes (ou colonnes) sont échangées, le déterminant change de signe.
 - Si une ligne (ou colonne) est multipliée par un scalaire k , le déterminant est multiplié par k .
 - Si une ligne (ou colonne) est ajoutée à une autre ligne (ou colonne), le déterminant reste inchangé.
- **Linéarité** : Le déterminant est linéaire par rapport à chaque ligne (ou colonne) lorsqu'on considère les autres lignes (ou colonnes) comme fixes.

7.6 Interprétation Géométrique du Déterminant

Le déterminant a une interprétation géométrique importante : il représente le **facteur de mise à l'échelle** (scaling) du volume lors de la transformation linéaire représentée par la matrice.

7.6.1 Exemple Numérique en 2D

Considérons la matrice :

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

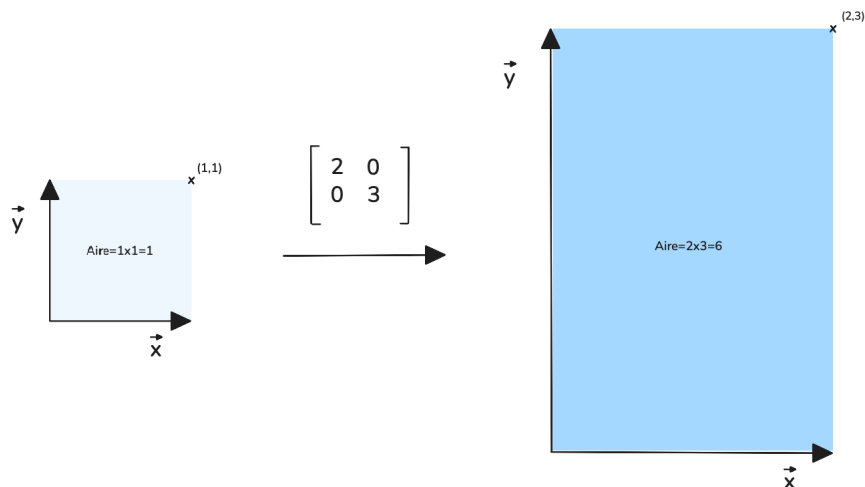
Le déterminant est $\det(A) = 2 \times 3 = 6$.

Appliquons la matrice A aux vecteurs de base e_1 et e_2 :

$$A \cdot e_1 = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$A \cdot e_2 = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

Ainsi, la matrice A étire l'axe x par un facteur de 2 et l'axe y par un facteur de 3. Cela confirme que l'aire du carré unité est multipliée par le déterminant $\det(A) = 6$.



7.6.2 Exemple Numérique en 3D

Considérons la matrice :

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

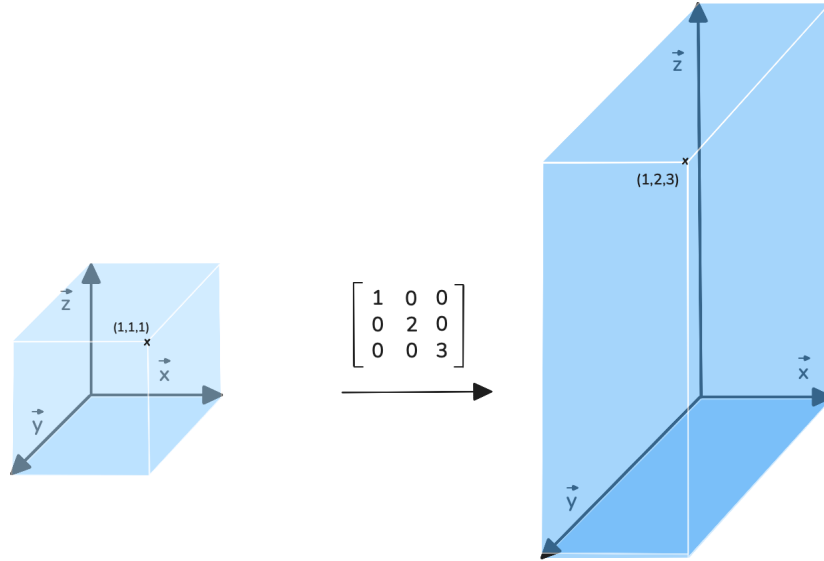
Le déterminant est $\det(A) = 1 \times 2 \times 3 = 6$.

Cette matrice étire les axes x , y et z par des facteurs de 1, 2 et 3 respectivement.

$$A \cdot e_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \times 1 + 0 \times 0 + 0 \times 0 \\ 0 \times 1 + 2 \times 0 + 0 \times 0 \\ 0 \times 1 + 0 \times 0 + 3 \times 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$A \cdot e_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \times 0 + 0 \times 1 + 0 \times 0 \\ 0 \times 0 + 2 \times 1 + 0 \times 0 \\ 0 \times 0 + 0 \times 1 + 3 \times 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}$$

$$A \cdot e_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \times 0 + 0 \times 0 + 0 \times 1 \\ 0 \times 0 + 2 \times 0 + 0 \times 1 \\ 0 \times 0 + 0 \times 0 + 3 \times 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}$$



Le volume d'un cube unité devient 6 après transformation.

7.6.3 Déterminant Nul et Aplatissement de l'Espace

Un déterminant nul indique que la transformation **écrase l'espace** dans une dimension inférieure. Cela signifie que le volume (ou l'aire en 2D) après transformation est nul.

Exemple Considérons la matrice :

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

Le déterminant est $\det(A) = 1 \times 4 - 2 \times 2 = 4 - 4 = 0$.

Cette matrice transforme l'espace en l'écrasant sur une ligne, réduisant l'aire à zéro. En effet, les deux vecteurs résultants sont colinéaires.

Calcul des Transformations des Axes

Appliquons la matrice A aux vecteurs de base e_1 et e_2 :

$$A \cdot e_1 = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \times 1 + 2 \times 0 \\ 2 \times 1 + 4 \times 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$A \cdot e_2 = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \times 0 + 2 \times 1 \\ 2 \times 0 + 4 \times 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

Observons les vecteurs résultants :

$$A \cdot e_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad A \cdot e_2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

Nous constatons que :

$$A \cdot e_2 = 2 \times \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 2 \times (A \cdot e_1)$$

Cela démontre que les vecteurs $A \cdot e_1$ et $A \cdot e_2$ sont colinéaires, c'est-à-dire qu'ils sont alignés sur la même ligne droite. Donc l'espace a été aplati en une dimension inférieure.

7.7 Pseudo-Déterminant

Le déterminant d'une matrice carrée offre des informations précieuses sur les propriétés de cette matrice. Un déterminant nul indique que la matrice est **singulière**, c'est-à-dire qu'elle n'est pas inversible. Cette condition est souvent liée à des problèmes tels que la dépendance linéaire entre les colonnes de la matrice.

Conséquences d'un Déterminant Nul

Quand une matrice a un déterminant nul, cela signifie qu'au moins une des dimensions de l'espace est "écrasée" lors de la transformation linéaire représentée par la matrice. En termes de systèmes d'équations, cela implique que le système peut ne pas avoir de solution unique, ou ne pas avoir de solution du tout.

Le Pseudo-Déterminant

Le pseudo-déterminant est une généralisation du déterminant pour les matrices singulières (c'est-à-dire les matrices dont le déterminant est nul). Il est défini comme le produit des valeurs singulières non nulles de la matrice. Pour une matrice A de rang r ayant r valeurs singulières non nulles $\sigma_1, \sigma_2, \dots, \sigma_r$, le pseudo-déterminant est donné par :

$$\text{pdet}(A) = \prod_{i=1}^r \sigma_i$$

En d’autres termes, le pseudo-déterminant prend en compte uniquement les dimensions non écrasées de la transformation, offrant une mesure utile lorsque le déterminant traditionnel ne peut pas être utilisé en raison de la singularité de la matrice.

Utilisation en Pratique

Dans les applications pratiques, notamment en machine learning avec des bibliothèques comme `scikit-learn`, le pseudo-déterminant est utilisé pour éviter les problèmes numériques liés aux matrices singulières. Par exemple, au lieu de calculer directement un déterminant qui pourrait être nul et provoquer des erreurs, des méthodes basées sur la **décomposition en valeurs singulières (SVD)** sont employées pour évaluer la “taille” de l’information contenue dans les données sans les problèmes numériques associés au calcul du déterminant classique.

7.8 Interprétation du Déterminant en Machine Learning

En machine learning, calculer le déterminant de la matrice représentant un dataset (les variables sont rangées par colonne) revient à évaluer l’indépendance des caractéristiques (features) représentées par les colonnes de la matrice.

Indicateur de Multicollinéarité

Un déterminant proche de zéro indique une forte **multicollinéarité** entre les variables du modèle (le rang est inférieur au nombre de variables). Cela signifie que certaines variables sont presque parfaitement linéairement dépendantes des autres. Dans les techniques de régression, par exemple, cela peut entraîner des problèmes de stabilité et d’interprétabilité des coefficients estimés. Le déterminant sert donc de diagnostic pour prévenir des problèmes potentiels dans les modèles prédictifs.

Importance du Pseudo-Déterminant

Comme mentionné précédemment, dans le cas de matrices singulières ou presque singulières, où le déterminant classique serait problématique (proche de zéro ou nul), l’utilisation du pseudo-déterminant offre une alternative pour évaluer la “taille” de l’information contenue dans les données sans les erreurs numériques associées au calcul du déterminant classique.

8 Inverse d'une matrice

8.1 Vers la résolution des équations matricielles

Nous nous dirigeons vers la résolution d'équations matricielles, similaires aux équations classiques (par exemple, résoudre pour x dans $4x = 8$), mais impliquant des matrices. Dans le cadre de la science des données, cette complexité est essentielle, car résoudre des équations matricielles est fondamental, notamment pour ajuster des modèles statistiques aux données (comme les modèles linéaires et la régression).

À la fin de ce chapitre, vous comprendrez ce qu'est l'inverse d'une matrice, quand elle peut être calculée, comment la calculer et comment l'interpréter.

8.2 L'Inverse de la Matrice

L'inverse d'une matrice A est une autre matrice, notée A^{-1} , telle que $A^{-1}A = I$, où I est la matrice identité. Cela permet de "réduire à l'identité" une matrice. On peut aussi voir cela comme une transformation linéaire ramenant la matrice A à l'identité via l'inverse.

Pourquoi inverser une matrice ? Principalement pour résoudre des équations de la forme $Ax = b$, où A et b sont connues, et où l'on cherche x . La solution générale est :

$$Ax = b \Rightarrow A^{-1}Ax = A^{-1}b \Rightarrow Ix = A^{-1}b \Rightarrow x = A^{-1}b$$

8.3 Types d'Inverses et Conditions d'Invertibilité

"Inverser une matrice" semble simple, mais toutes les matrices ne sont pas inversibles. Il existe trois types d'inverses, chacun avec des conditions spécifiques d'invertibilité.

8.3.1 Inverse complet

Cela signifie que $A^{-1}A = AA^{-1} = I$. Deux conditions sont nécessaires pour qu'une matrice ait un inverse complet : elle doit être (1) carrée et (2) de rang complet. Toute matrice carrée de rang complet possède un inverse complet. Dans la suite, l'inverse complet sera simplement désigné comme l'"inverse".

Pour une matrice carrée, avoir un déterminant non nul ($\det(A) \neq 0$) est équivalent au fait que la matrice est de rang complet et donc inversible. En effet, A est inversible si et seulement si $\det(A) \neq 0$.

On remarquera que si A est inversible alors $\det(A^{-1}) = \frac{1}{\det(A)}$.

8.3.2 Inverse unilatéral

Une matrice rectangulaire peut avoir un inverse unilatéral, valable pour un seul ordre de multiplication. Par exemple, une matrice "haute" T peut avoir un **inverse gauche** L tel que $LT = I$, mais $TL \neq I$. Une matrice "large" W peut avoir un **inverse droit** R tel que $WR = I$, mais $RW \neq I$. Une matrice non carrée possède un inverse unilatéral uniquement si elle est de rang maximal.

8.3.3 Pseudo-inverse

Toute matrice a un pseudo-inverse, quelle que soit sa forme ou son rang. Si la matrice est carrée et de rang complet, son pseudo-inverse est égal à son inverse complet. Si elle est non carrée et de rang maximal, le pseudo-inverse est égal à l'inverse gauche (pour une matrice "haute") ou à l'inverse droit (pour une matrice "large"). Pour une matrice de rang réduit, le pseudo-inverse transforme la matrice en une approximation de la matrice identité.

Les matrices qui n'ont ni inverse complet ni inverse unilatéral sont appelées **singulières** ou **non inversibles**, aussi qualifiées de **rang réduit** ou **déficientes en rang**. Pour une matrice carrée, cela se traduit par un déterminant nul ($\det(A) = 0$), ce qui implique que les colonnes (ou lignes) de la matrice sont linéairement dépendantes.

Remarque importante : En pratique, on évite de calculer directement l'inverse gauche ou droit car cela est numériquement instable et sensible aux erreurs. À la place, on utilise le **pseudo-inverse** (calculé avec la décomposition SVD via `np.linalg.pinv`) ou la **décomposition QR** (gérée par `np.linalg.lstsq` dans NumPy), qui sont plus stables et adaptées aux matrices mal conditionnées ou de rang réduit. Ici on verra les inverses unilatéraux pour l'exemple.

8.4 Calcul de l'inverse

L'inverse d'une matrice semble intéressant, mais comment le calculer ? En prenant l'exemple des nombres scalaires, l'inverse de 3 est $\frac{1}{3}$, soit 3^{-1} . On pourrait penser qu'il suffit d'inverser chaque élément de la matrice :

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \begin{bmatrix} 1/a & 1/b \\ 1/c & 1/d \end{bmatrix}$$

Malheureusement, cette approche ne donne pas la matrice identité. En multipliant l'originale par cette "inverse" supposée, on obtient :

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1/a & 1/b \\ 1/c & 1/d \end{bmatrix} = \begin{bmatrix} 1 + \frac{b}{c} & \frac{a}{b} + \frac{b}{d} \\ \frac{c}{a} + \frac{d}{c} & 1 + \frac{c}{b} \end{bmatrix}$$

Ce calcul est valide mais ne produit pas la matrice identité. Donc, inverser chaque élément n'est pas la solution.

8.4.1 Inverse d'une matrice 2×2

Pour inverser une matrice 2×2 , on échange les éléments diagonaux, on change le signe des éléments hors diagonale, puis on divise par le déterminant :

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \Rightarrow A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Ainsi, on obtient :

$$AA^{-1} = \frac{1}{ad - bc} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Exemple numérique

$$\begin{bmatrix} 1 & 4 \\ 2 & 7 \end{bmatrix} \begin{bmatrix} 7 & -4 \\ -2 & 1 \end{bmatrix} \frac{1}{7-8} = \begin{bmatrix} (7-8) & (-4+4) \\ (14-14) & (-8+7) \end{bmatrix} \frac{1}{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
import numpy as np

A = np.array([[1, 4], [2, 7]])
# On calcule l'inverse de A
Ainv = np.linalg.inv(A)
# On multiplie A par son inverse pour vérification
A @ Ainv
```

On vérifie que $A @ Ainv$ donne bien la matrice identité. Attention, $A * Ainv$ ne donne pas la matrice identité car $*$ correspond à la multiplication élémentaire (Hadamard).

La matrice, son inverse et leur produit est visualisé ci-dessous :

A	A^{-1}	$A^{-1}A$
1 4	-7.0 4.0	1.0 0.0
2 7	2.0 -1.0	0.0 1.0

Essayons un autre exemple. Considérons la matrice suivante :

$$\begin{bmatrix} 1 & 4 \\ 2 & 8 \end{bmatrix} \begin{bmatrix} 8 & -4 \\ -2 & 1 \end{bmatrix} \frac{1}{0} = \begin{bmatrix} (8-8) & (-4+4) \\ (16-16) & (-8+8) \end{bmatrix} \frac{1}{0} = ???$$

Il y a plusieurs problèmes ici. Le déterminant est nul, donc on ne peut pas diviser par zéro, ce qui rend la matrice non inversible.

Qu'est-ce qui différencie cet exemple ? C'est une matrice de rang réduit (rang = 1). Cela montre qu'une matrice de rang réduit n'est pas inversible.

En Python, cela donne l'erreur suivante :

```
A = np.array([[1,4],[2,8]])
Ainv = np.linalg.inv(A)
```

Résultat : `LinAlgError: Singular matrix`

8.4.2 Inverse d'une Matrice Diagonale

Il existe une astuce pour calculer l'inverse d'une matrice diagonale carrée : on inverse simplement chaque élément de la diagonale et on ignore les zéros en dehors de la diagonale.

Exemple :

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} b & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & d \end{bmatrix} = \begin{bmatrix} 2b & 0 & 0 \\ 0 & 3c & 0 \\ 0 & 0 & 4d \end{bmatrix}$$

Pour obtenir l'inverse, il suffit de définir $b = \frac{1}{2}$, $c = \frac{1}{3}$ et $d = \frac{1}{4}$.

Que se passe-t-il si la matrice diagonale contient un zéro sur la diagonale ? On ne peut pas inverser cet élément, car cela donnerait une division par zéro. Ainsi, une matrice diagonale avec au moins un zéro sur la diagonale n'a pas d'inverse. Notez également qu'une matrice diagonale est de rang complet uniquement si tous les éléments de la diagonale sont non nuls.

L'inverse d'une matrice diagonale est important car il mène directement à la formule pour le calcul du pseudo-inverse. Nous y reviendrons plus tard.

8.4.3 Inversion d'une Matrice Carrée de Rang Complet

Le calcul de l'inverse d'une matrice inversible est un processus long et complexe, et il est rare que vous ayez besoin de le faire manuellement. Dans la pratique, on utilise `“np.linalg.inv”` ou d'autres fonctions similaires en Python.

8.4.4 Inverse gauche

Une matrice "haute" (de taille $M > N$) n'a pas d'inverse complet. En d'autres termes, pour une matrice T de taille $M \times N$, il n'existe pas de matrice T^{-1} telle que $TT^{-1} = T^{-1}T = I$.

Cependant, il existe une matrice L telle que $LT = I$. Notre objectif est maintenant de trouver cette matrice. Pour cela, on commence par transformer T en une matrice carrée. Comment rendre une matrice non carrée en carrée ? On la multiplie par sa transposée.

La question suivante est : devons-nous calculer $T^T T$ ou TT^T ? Les deux sont carrées, mais $T^T T$ est de rang complet si T est de rang colonne complet. Pourquoi est-ce important ? Car toute matrice carrée de rang complet possède un inverse.

Avant de présenter la dérivation de l'inverse gauche, démontrons en Python qu'une matrice " haute " multipliée par sa transposée a un inverse complet :

```
import numpy as np

T = np.random.randint(-10, 11, size=(40, 4))
TtT = T.T @ T
TtT_inv = np.linalg.inv(TtT)
TtT_inv @ TtT # devrait produire l'identité
```

Cela peut se traduire mathématiquement par :

$$(T^T T)^{-1} (T^T T) = I$$

En examinant le code et la formule, on voit que, puisque $T^T T$ n'est pas la même matrice que T , $(T^T T)^{-1}$ n'est pas l'inverse de T .

Cependant, notre objectif est de trouver une matrice qui, multipliée à gauche par T , produit la matrice identité. Peu importe quelles autres matrices doivent être multipliées pour obtenir cette matrice. La solution est donc de regrouper les multiplications de la manière suivante :

$$L = (T^T T)^{-1} T^T$$
$$LT = I$$

Ainsi, L est l'inverse gauche de T .

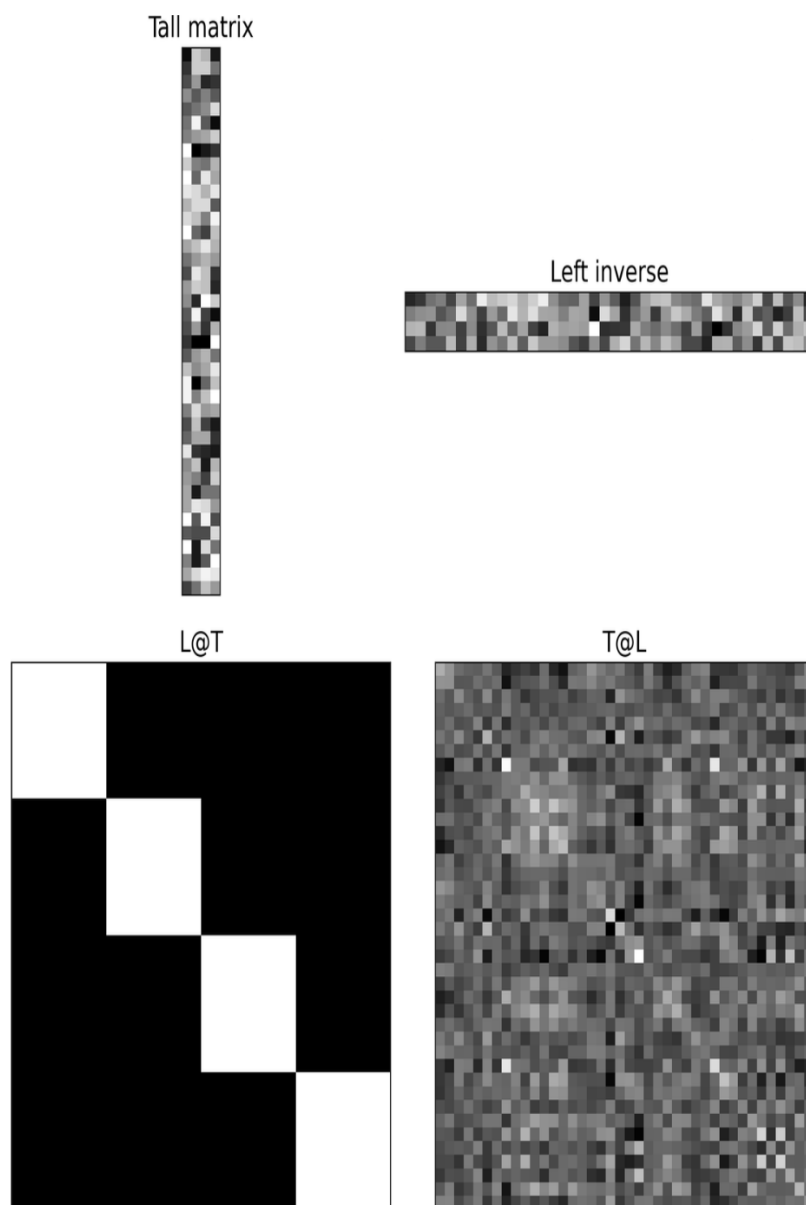
En Python, on peut finaliser le code pour calculer l'inverse gauche et vérifier qu'il produit bien la matrice identité en multipliant à gauche :

```
import numpy as np

L = TtT_inv @ T.T # inverse gauche
L @ T # produit la matrice identité
```

On peut également vérifier que TL (c'est-à-dire en multipliant à droite par l'inverse gauche) **ne** donne **pas** la matrice identité. C'est pourquoi l'inverse gauche est un inverse unilatéral.

L'image ci-dessous montre une matrice haute, son inverse gauche et les deux moyens de multiplier la matrice inverse gauche par la matrice.



8.4.5 Inverse droit

Pour une matrice "large" (de taille $M < N$), il n'existe pas d'inverse complet, mais il est possible de définir un **inverse droit**. Pour une matrice W de taille large, l'inverse droit R est défini tel que $WR = I$.

Pour obtenir cet inverse droit, on peut utiliser la transposée pour créer une matrice carrée en multipliant W par sa transposée. On calcule alors :

$$R = W^T(WW^T)^{-1}$$

En Python, cela peut être implémenté comme suit :

```
import numpy as np

# Générer une matrice large W (par exemple, taille 4x40)
W = np.random.randint(-10, 11, size=(4, 40))

# Calculer W * W^T et son inverse
WWt = W @ W.T
WWt_inv = np.linalg.inv(WWt)

# Calculer l'inverse droit
R = W.T @ WWt_inv

# Vérifier que WR produit la matrice identité
identity_check = W @ R
print(identity_check)
```

En mathématiques, cela revient à :

$$WR = I$$

Cet inverse droit R est défini uniquement pour les matrices "larges" de rang ligne complet, c'est-à-dire lorsque W a un rang maximum égal au nombre de lignes.

8.4.6 Pseudo-Inverse de Moore-Penrose

Comme mentionné précédemment, il est impossible de transformer une matrice de rang réduit en matrice identité par simple multiplication matricielle. Cela signifie que les matrices de rang réduit n'ont pas d'inverse complet ou unilatéral. Cependant, les matrices singulières possèdent un **pseudo-inverse**. Les pseudo-inverses sont des matrices de transformation qui amènent une matrice à se rapprocher de la matrice identité.

Le terme **pseudo-inverses** (au pluriel) n'est pas une erreur. Bien que l'inverse complet soit unique, le pseudo-inverse ne l'est pas. Une matrice de rang réduit possède un nombre infini de pseudo-inverses. Toutefois, il existe un pseudo-inverse particulier, le **pseudo-inverse de Moore-Penrose**, qui est le plus utilisé et donc généralement désigné simplement par **pseudo-inverse**.

Le pseudo-inverse de Moore-Penrose est noté A^\dagger , A^+ ou A^* . En Python, il est calculé avec la fonction “`np.linalg.pinv`”. Voici un exemple de code pour calculer le pseudo-inverse d'une matrice singulière :

```
import numpy as np

A = np.array([[1, 4], [2, 8]])
Apinv = np.linalg.pinv(A)
A @ Apinv # Cette opération donne une matrice proche de
           l'identité
```

Comment le pseudo-inverse est-il calculé ? L'algorithme pour calculer le pseudo-inverse repose sur la **décomposition en valeurs singulières** (SVD). Pour le calculer, on effectue la SVD de la matrice, on inverse les valeurs singulières non nulles sans modifier les vecteurs singuliers, puis on recompose la matrice en multipliant $U\Sigma^+V^T$.

8.5 L'Inverse complet est Unique

L'inverse d'une matrice est unique, ce qui signifie que si une matrice a un inverse, elle n'en a qu'un seul. Il ne peut pas y avoir deux matrices B et C telles que $AB = I$ et $AC = I$ tout en ayant $B \neq C$.

Il existe plusieurs démonstrations de cette affirmation. Ici, on utilise une technique appelée **preuve par négation**. Cela consiste à essayer de prouver une fausse affirmation pour démontrer la véracité de l'affirmation d'origine. Dans ce cas, on part des hypothèses suivantes : (1) la matrice A est inversible, (2) les matrices B et C sont inverses de A , et (3) les matrices B et C sont distinctes, ce qui signifie $B \neq C$.

Suivez chaque expression de gauche à droite en notant que chaque expression repose sur l'ajout ou la suppression de la matrice identité, exprimée en tant que matrice multipliée par son inverse :

$$C = CI = CAB = IB = B$$

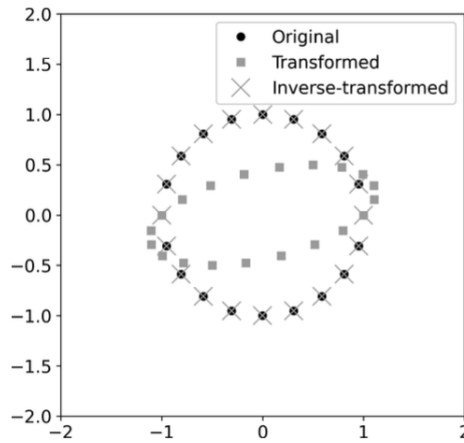
Toutes les expressions sont égales, ce qui signifie que la première et la dernière sont également égales. Cela implique que notre hypothèse $B \neq C$ est fausse. La conclusion est que deux matrices prétendant être l'inverse de la même matrice

sont égales. En d'autres termes, une matrice inversible a exactement un seul inverse.

8.6 Interprétation Géométrique de l'Inverse

On rappelle que la multiplication matrice-vecteur est vue comme une transformation géométrique d'un vecteur ou d'un ensemble de points.

Dans cette perspective, on peut considérer l'inverse d'une matrice comme annulant la transformation géométrique imposée par la multiplication matricielle. La Figure ci-dessous montre un exemple où les coordonnées géométriques transformées sont ramenées à leur état d'origine en multipliant par l'inverse de la matrice de transformation.



Cet effet géométrique est conforme aux mathématiques sous-jacentes. Dans les équations ci-dessous, P est la matrice $2 \times N$ des coordonnées géométriques d'origine, T est la matrice de transformation, Q est la matrice des coordonnées transformées, et U est la matrice des coordonnées après application de l'inverse :

$$Q = TP$$

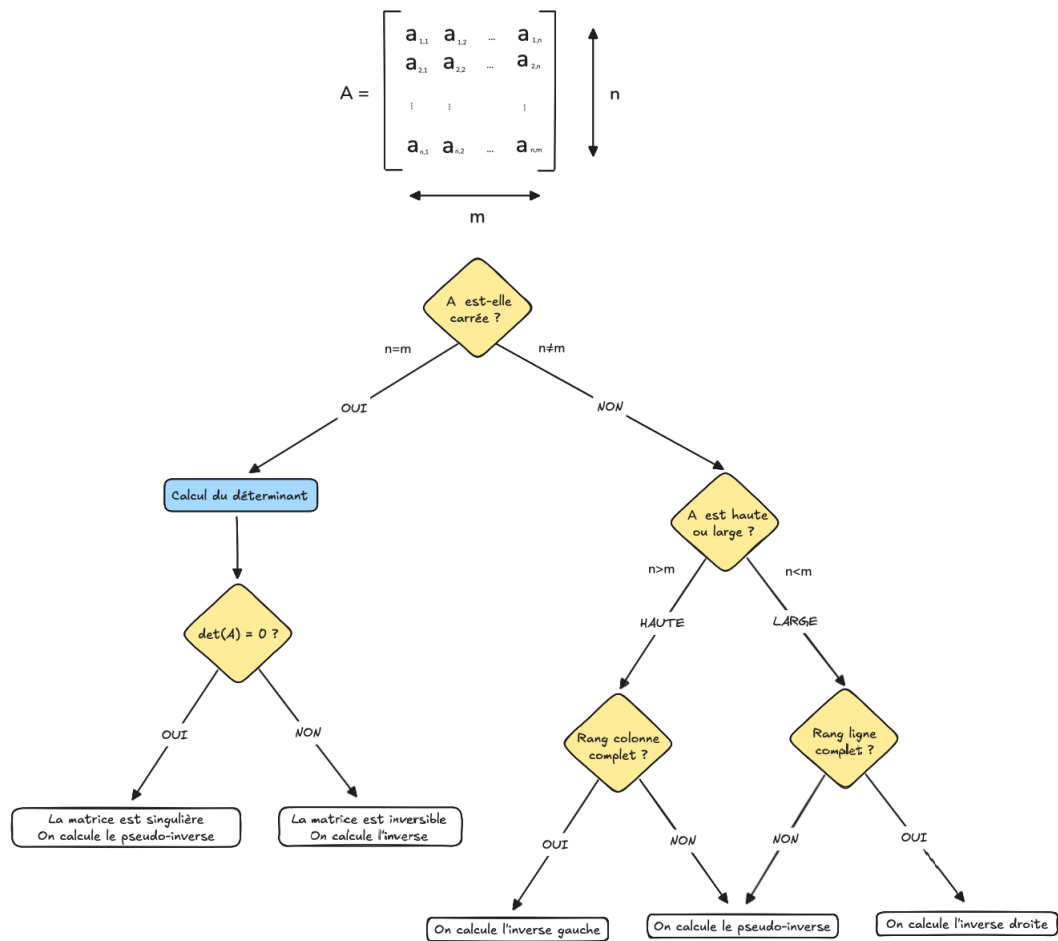
$$U = T^{-1}Q$$

$$U = T^{-1}TP$$

Bien que ce résultat ne soit pas surprenant, il aide à comprendre l'utilité géométrique de l'inverse d'une matrice : **annuler la transformation imposée par la matrice**. Cette interprétation sera utile lors de l'apprentissage de la diagonalisation d'une matrice par décomposition en valeurs propres.

Cette interprétation géométrique fournit également une intuition sur la raison pour laquelle une matrice de rang réduit n'a pas d'inverse : l'effet géométrique de la transformation par une matrice singulière est de "compresser" au moins une dimension. Une fois qu'une dimension est aplatie, elle ne peut pas être rétablie, tout comme vous ne pouvez pas voir votre dos en vous tenant devant un miroir.

8.7 Résumé des situations



8.8 Application : Résolution d'une Équation Matricielle

La résolution de l'équation matricielle $AX = B$ est une application directe de l'inversion de matrices et est étroitement liée à la résolution de systèmes d'équations linéaires. Ici, A est une matrice connue, B est un vecteur ou une matrice connue, et X est le vecteur ou la matrice inconnue que nous souhaitons déterminer.

8.8.1 Lien avec les Systèmes d'Équations Linéaires

L'équation $AX = B$ représente un système d'équations linéaires. Chaque ligne de la matrice A correspond aux coefficients d'une équation, chaque composante de X est une variable inconnue, et chaque composante de B est le terme indépendant de l'équation correspondante. Résoudre $AX = B$ revient donc à trouver les valeurs des variables qui satisfont toutes les équations simultanément.

8.8.2 Résolution à l'Aide de l'Inverse d'une Matrice

Si A est une matrice carrée et inversible, nous pouvons résoudre $AX = B$ en utilisant l'inverse de A .

Importance de l'Ordre des Multiplications

Il est important de rappeler que la multiplication des matrices n'est **pas commutative**, c'est-à-dire que $AB \neq BA$ en général. Par conséquent, pour isoler X , nous multiplions à gauche par A^{-1} :

$$AX = B \implies A^{-1}AX = A^{-1}B \implies IX = A^{-1}B \implies X = A^{-1}B$$

8.8.3 Exemple Numérique

Cas où A est inversible :

Soit le système d'équations linéaires suivant :

$$\begin{cases} 2x_1 + x_2 = 5 \\ 3x_1 + 4x_2 = 11 \end{cases}$$

Nous pouvons écrire ce système sous forme matricielle :

$$A = \begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad B = \begin{pmatrix} 5 \\ 11 \end{pmatrix}$$

Nous souhaitons résoudre $AX = B$ pour X .

Étapes :

1. Vérifier que A est inversible :

Calculons le déterminant de A :

$$\det(A) = 2 * 4 - 1 * 3 = 8 - 3 = 5 \neq 0$$

Donc, A est inversible.

2. Calculer l'inverse de A :

$$A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} 4 & -1 \\ -3 & 2 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 4 & -1 \\ -3 & 2 \end{pmatrix}$$

3. Calculer $X = A^{-1}B$:

$$X = A^{-1}B = \frac{1}{5} \begin{pmatrix} 4 & -1 \\ -3 & 2 \end{pmatrix} \begin{pmatrix} 5 \\ 11 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 4 * 5 + (-1) * 11 \\ (-3) * 5 + 2 * 11 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 20 - 11 \\ -15 + 22 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 9 \\ 7 \end{pmatrix}$$

Donc,

$$X = \begin{pmatrix} \frac{9}{5} \\ \frac{7}{5} \end{pmatrix} = \begin{pmatrix} 1.8 \\ 1.4 \end{pmatrix}$$

Interprétation :

Nous avons trouvé les valeurs des variables :

$$x_1 = 1.8, \quad x_2 = 1.4$$

Vérification :

Substituons ces valeurs dans les équations initiales :

- Première équation :

$$2 * 1.8 + 1.4 = 3.6 + 1.4 = 5 \quad (\text{OK})$$

- Deuxième équation :

$$3 * 1.8 + 4 * 1.4 = 5.4 + 5.6 = 11 \quad (\text{OK})$$

Ainsi, la solution est correcte.

8.8.4 Cas où A n'est pas inversible ou n'est pas carrée

Lorsque A n'est pas inversible (par exemple, son déterminant est nul) ou n'est pas carrée (nombre de lignes différent du nombre de colonnes), nous ne pouvons pas utiliser l'inverse classique de A . Dans ce cas, nous pouvons utiliser le **pseudo-inverse de Moore-Penrose**, noté A^+ , pour trouver une solution approchée.

Exemple :

Soit le système surdéterminé (plus d'équations que d'inconnues) :

$$\begin{cases} x_1 + 2x_2 = 5 \\ 3x_1 + 4x_2 = 6 \\ 5x_1 + 6x_2 = 7 \end{cases}$$

Sous forme matricielle :

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad B = \begin{pmatrix} 5 \\ 6 \\ 7 \end{pmatrix}$$

Ici, A est une matrice 3×2 , donc non carrée.

Méthode de résolution :

Nous utilisons le pseudo-inverse de Moore-Penrose :

$$X = A^+ B$$

Calcul du pseudo-inverse avec NumPy :

```
import numpy as np

A = np.array([[1, 2],
              [3, 4],
              [5, 6]])
B = np.array([5, 6, 7])

# Calcul du pseudo-inverse de Moore-Penrose
A_pinv = np.linalg.pinv(A)

# Calcul de X
X = A_pinv @ B

print("Solution X :", X)
```


Résultat :

La solution X obtenue est la meilleure approximation possible compte tenu des données.

8.8.5 Interprétation

L'utilisation de l'inverse (classique ou pseudo-inverse) pour résoudre $AX = B$ permet de généraliser la résolution des systèmes d'équations linéaires, qu'ils soient déterminés, surdéterminés ou sous-déterminés.

- Si le système est déterminé (nombre d'équations égal au nombre d'inconnues) et A est inversible, la solution est unique.
- Si le système est surdéterminé (plus d'équations que d'inconnues), il n'y a généralement pas de solution exacte, mais la pseudo-inverse fournit la solution qui minimise l'erreur globale.
- Si le système est sous-déterminé (moins d'équations que d'inconnues), il existe une infinité de solutions, et la pseudo-inverse donne une solution particulière de moindre norme.

9 Éléments propres et polynôme caractéristique

9.1 Vecteurs et valeurs propres

Soit A une matrice carrée. Un **vecteur propre** de A est un vecteur non nul \vec{v} tel que

$$A\vec{v} = \lambda\vec{v},$$

où λ est un scalaire. On dit alors que \vec{v} est un vecteur propre de A associé à la valeur propre λ et que le sous-espace vectoriel $E_\lambda = \{\vec{v} \in \mathbb{K}^n, A\vec{v} = \lambda\vec{v}\}$ est le sous-espace propre associé à λ .

Exemple : Soit la matrice

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

En remarquant que $M * \begin{pmatrix} 4 \\ 3 \\ -2 \end{pmatrix} = 2 * \begin{pmatrix} 4 \\ 3 \\ -2 \end{pmatrix}$

Alors on peut dire que le vecteur propre associé à la valeur propre $\lambda = 2$ est $(4, 3, -2)$.

9.2 Polynôme caractéristique

Définition : Soit $A \in \mathcal{M}_n(\mathbb{K})$. On appelle **polynôme caractéristique** de A le polynôme :

$$\chi_A(X) = \det(A - XI_n).$$

Remarque : en dimension 2 on a :

$$\chi_A(X) = X^2 - \text{tr}(A)X + \det(A).$$

Exemple : Soit la matrice

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

Pour trouver le polynôme caractéristique de M , on calcule $\det(M - XI_3)$, où I_3 est la matrice identité de dimension 3 :

$$M - XI = \begin{pmatrix} -X & 2 & -1 \\ 3 & -2 - X & 0 \\ -2 & 2 & 1 - X \end{pmatrix}.$$

Le déterminant de cette matrice est :

$$\det(M - \lambda I) = \begin{vmatrix} -X & 2 & -1 \\ 3 & -2 - X & 0 \\ -2 & 2 & 1 - X \end{vmatrix}.$$

En développant ce déterminant par la 3ème colonne, on obtient :

$$\begin{aligned} \det(M - XI) &= -1 \begin{vmatrix} 3 & -2 - X \\ -2 & 2 \end{vmatrix} + (1 - X) \begin{vmatrix} -X & 2 \\ 3 & -2 - X \end{vmatrix} \\ &= (-1)(6 - 4 - 2X) + (1 - X)(2X + X^2 - 6) \\ &= (-2)(1 - X) + (1 - X)(X^2 + 2X - 6) \\ &= (1 - X)(X^2 + 2X - 8) \\ &= (1 - X)(X + 4)(X - 2) \end{aligned}$$

Le polynôme caractéristique de M est donc :

$$\chi_M(X) = (1 - X)(X + 4)(X - 2).$$

On veillera à toujours garder une forme factoriser au maximum pour facilité la recherche des racines.

9.3 Calcul des valeurs propres et des vecteurs propres

Méthode

Pour trouver les valeurs propres d'une matrice A , on résout le **polynôme caractéristique**, obtenu en égalant à zéro le déterminant de $A - \lambda I$, où I est la matrice identité de dimension appropriée :

$$\det(A - \lambda I) = 0.$$

Les solutions λ de cette équation sont les valeurs propres. Pour chaque valeur propre λ , les vecteurs propres correspondants sont les solutions non triviales \vec{v} de l'équation

$$(A - \lambda I)\vec{v} = 0.$$

En d'autres termes, les valeurs propres de A sont les racines du polynôme caractéristique.

L'ensemble des valeurs propres d'une matrice est appelé le **spectre** de A et est noté $sp(A)$.

En reprenant l'exemple précédent avec $\chi_M(X) = (1 - X)(X + 4)(X - 2)$, on en déduit que $sp(M) = \{-4, 1, 2\}$.

Si on veut par exemple trouver le vecteur propre associé à la valeur propre -4 , alors on pose $\vec{v} = (x_1, x_2, x_3)$ un vecteur de \mathbb{R}^3 , et on résout $M\vec{v} = -4\vec{v}$. On obtient donc :

$$(M + 4I)\vec{v} = \begin{pmatrix} 4 & 2 & -1 \\ 3 & 2 & 0 \\ -2 & 2 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \vec{0}.$$

Nous devons donc résoudre le système homogène suivant :

$$\begin{cases} 4x_1 + 2x_2 - x_3 = 0 \\ 3x_1 + 2x_2 = 0 \\ -2x_1 + 2x_2 + 5x_3 = 0 \end{cases}.$$

Résolvons ce système :

1. À partir de la deuxième équation :

$$3x_1 + 2x_2 = 0 \implies x_2 = -\frac{3}{2}x_1.$$

2. Substituons x_2 dans la première équation :

$$4x_1 + 2\left(-\frac{3}{2}x_1\right) - x_3 = 0 \implies 4x_1 - 3x_1 - x_3 = 0 \implies x_1 - x_3 = 0 \implies x_3 = x_1.$$

3. Substituons x_2 et x_3 dans la troisième équation pour vérification :

$$-2x_1 + 2\left(-\frac{3}{2}x_1\right) + 5x_1 = -2x_1 - 3x_1 + 5x_1 = 0,$$

ce qui est vérifié.

Ainsi, le vecteur propre associé à la valeur propre -4 est donné par $\vec{v} = (x_1, x_2, x_3) = x_1(1, -\frac{3}{2}, 1)$.

Pour simplifier, nous pouvons écrire le vecteur propre comme :

$$\vec{v} = k \begin{pmatrix} 1 \\ -\frac{3}{2} \\ 1 \end{pmatrix} \quad \text{pour tout scalaire } k \neq 0.$$

```
import numpy as np

# Définition de la matrice M
M = np.array([[0, 2, -1],
              [3, -2, 0],
              [-2, 2, 1]])

# Calcul des valeurs propres et des vecteurs propres
valeurs_propres, vecteurs_propres = np.linalg.eig(M)

print("Valeurs propres de M :", valeurs_propres)
print("Vecteurs propres de M :\n", vecteurs_propres)
```

Remarque : Si vous ne trouvez pas les mêmes vecteurs propres c'est normal, car numpy les normalise.

9.4 Propriétés

- Une matrice de taille $n \times n$ a au plus n valeurs propres distinctes.
- Les vecteurs propres correspondant à des valeurs propres distinctes sont linéairement indépendants.
- Si une matrice A est diagonale, alors ses valeurs propres sont simplement les éléments sur la diagonale principale de A .
- Le polynôme caractéristique d'une matrice carrée de taille n est unitaire de degré n .
- Une matrice et sa transposée ont le même polynôme caractéristique. (donc les mêmes valeurs propres, et avec les mêmes ordres de multiplicité)

9.5 Multiplicité des valeurs propres

Définitions :

- La **multiplicité algébrique** d'une valeur propre λ est son ordre de multiplicité en tant que racine du polynôme caractéristique $\chi_A(\lambda)$.
- La **multiplicité géométrique** d'une valeur propre λ est la dimension du sous-espace propre E_λ .

Propriété : Pour chaque valeur propre λ de A , la multiplicité géométrique est toujours inférieure ou égale à la multiplicité algébrique :

$$1 \leq \dim(E_\lambda) \leq m,$$

où m est la multiplicité algébrique de λ .

Exemple :

Soit la matrice :

$$B = \begin{pmatrix} 5 & 4 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Calculons le polynôme caractéristique :

$$\chi_B(\lambda) = \det(B - \lambda I) = \begin{vmatrix} 5 - \lambda & 4 & 2 \\ 0 & 1 - \lambda & -1 \\ 0 & 0 & 1 - \lambda \end{vmatrix}.$$

Le déterminant se calcule facilement en utilisant la propriété des matrices triangulaires supérieures :

$$\chi_B(\lambda) = (5 - \lambda)(1 - \lambda)^2.$$

Ainsi, les valeurs propres sont $\lambda = 5$ (multiplicité algébrique 1) et $\lambda = 1$ (multiplicité algébrique 2).

Calculons le sous-espace propre associé à $\lambda = 1$:

$$B - I = \begin{pmatrix} 4 & 4 & 2 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Le système $(B - I)\vec{v} = \vec{0}$ donne :

$$\begin{cases} 4x_1 + 4x_2 + 2x_3 = 0, \\ -x_3 = 0, \\ 0 = 0. \end{cases}$$

De la deuxième équation, on obtient $x_3 = 0$. La première équation devient :

$$4x_1 + 4x_2 = 0 \implies x_1 + x_2 = 0 \implies x_1 = -x_2.$$

Le sous-espace propre E_1 est donc :

$$E_1 = \left\{ \begin{pmatrix} -x_2 \\ x_2 \\ 0 \end{pmatrix} \mid x_2 \in \mathbb{K} \right\}.$$

La dimension de E_1 est 1, alors que la multiplicité algébrique de $\lambda = 1$ est 2.

9.6 Lien avec la trace et le déterminant

Remarque: Polynôme scindé

Définition : Un polynôme est dit **scindé** sur un corps \mathbb{K} s'il peut être factorisé en produit de polynômes de degré 1 à coefficients dans \mathbb{K} . Autrement dit, il s'écrit sous la forme :

$$P(\lambda) = a(\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_k),$$

où $a \in \mathbb{K}$ et $\lambda_i \in \mathbb{K}$.

Exemple : le polynôme $P(\lambda) = (\lambda - 2)^2(\lambda + 1)$ est scindé sur $\mathbb{K} = \mathbb{R}$.

Propriété : Soit $A \in \mathcal{M}_n(\mathbb{K})$. On suppose que le polynôme caractéristique est scindé. Alors A admet exactement n valeurs propres $\lambda_1, \dots, \lambda_n$ comptées avec multiplicité, et on a :

$$\det(A) = \prod_{i=1}^n \lambda_i \quad \text{et} \quad \text{tr}(A) = \sum_{i=1}^n \lambda_i.$$

Exemple :

Reprenons la matrice :

$$B = \begin{pmatrix} 5 & 4 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Nous avons déjà déterminé que les valeurs propres de B sont :

- $\lambda_1 = 5$ avec une multiplicité algébrique de 1.
- $\lambda_2 = 1$ avec une multiplicité algébrique de 2.

Calculons la trace et le déterminant de B :

Calcul de la trace :

La trace d'une matrice est la somme de ses éléments diagonaux :

$$\text{tr}(B) = b_{11} + b_{22} + b_{33} = 5 + 1 + 1 = 7.$$

La somme des valeurs propres (en tenant compte des multiplicités) est :

$$\lambda_1 + 2\lambda_2 = 5 + 2 \times 1 = 7.$$

Nous constatons que :

$$\text{tr}(B) = \sum_{i=1}^n \lambda_i.$$

Calcul du déterminant :

Puisque B est une matrice triangulaire supérieure, son déterminant est le produit de ses éléments diagonaux :

$$\det(B) = b_{11} \times b_{22} \times b_{33} = 5 \times 1 \times 1 = 5.$$

Le produit des valeurs propres (en tenant compte des multiplicités) est :

$$\lambda_1 \times \lambda_2^2 = 5 \times (1)^2 = 5.$$

Ainsi, nous avons :

$$\det(B) = \prod_{i=1}^n \lambda_i.$$


```

import numpy as np

# Calcul de la trace et du déterminant de la matrice M
M = np.array([[0, 2, -1],
               [3, -2, 0],
               [-2, 2, 1]])

trace_M = np.trace(M)
det_M = np.linalg.det(M)

print("Trace de M :", trace_M)
print("Déterminant de M :", det_M)

# Calcul du produit des valeurs propres
valeurs_propres, _ = np.linalg.eig(M)
produit_valeurs_propres = np.prod(valeurs_propres)

print("Produit des valeurs propres :",
      produit_valeurs_propres)

# Calcul de la somme des valeurs propres
somme_valeurs_propres = np.sum(valeurs_propres)
print("Somme des valeurs propres :",
      somme_valeurs_propres)

# Vérification des propriétés
print("Trace égale à la somme des valeurs propres :", np.
      isclose(trace_M, somme_valeurs_propres))
print("Déterminant égal au produit des valeurs propres :",
      np.isclose(det_M, produit_valeurs_propres))

```

9.7 Matrice non inversible et valeur propre nulle

Propriété : Une matrice A est non inversible si et seulement si 0 est une valeur propre de A .

Démonstration : Si 0 est une valeur propre de A , alors il existe un vecteur non nul \vec{v} tel que $A\vec{v} = 0$. Donc, A n'est pas inversible (car \vec{v} est non nul !). Réciproquement, si A n'est pas inversible, alors $\det(A) = 0$. Comme le déterminant est le produit des valeurs propres, 0 est nécessairement une valeur propre de A . \square

9.7.1 Propriétés des Matrices Symétriques

On rappelle qu'une matrice symétrique est une matrice carrée égale à sa transposée.

Propriété : Les vecteurs propres d'une matrice symétrique réelle sont orthogonaux entre eux.

Autrement dit, si v et w sont des vecteurs propres associés à des valeurs propres distinctes de A , alors ils sont orthogonaux (produit scalaire nul) :

$$v^T w = 0.$$

Normalisation des vecteurs propres :

En normalisant ces vecteurs propres (c'est-à-dire en les rendant de norme unitaire) et en les arrangeant en colonnes dans une matrice Q , nous obtenons une matrice orthogonale :

$$Q = [v_1 \ v_2 \ \dots \ v_n], \quad \text{avec} \quad Q^T Q = I.$$

Conséquence : La matrice Q représente une rotation (ou une réflexion si le déterminant est négatif) dans l'espace vectoriel.

10 Diagonalisation d'une matrice

10.1 Définition

La diagonalisation est un processus qui transforme une matrice carrée en une matrice diagonale à l'aide d'un changement de base approprié. Une matrice A est dite **diagonalisable** s'il existe une base de \mathbb{K}^n formée de vecteurs propres de A . Dans cette base, la matrice de A est diagonale, c'est-à-dire que tous les coefficients hors diagonale sont nuls. Formellement, A est diagonalisable s'il existe une matrice inversible P telle que :

$$A = PDP^{-1},$$

où D est une matrice diagonale dont les éléments diagonaux sont les valeurs propres de A , et P est une matrice dont les colonnes sont les vecteurs propres correspondants.

```
import numpy as np

# Définition de la matrice A
A = np.array([[1, 0, 0],
              [1, 2, 0],
              [1, 0, 2]])

# Calcul des valeurs propres et des vecteurs propres
valeurs_propres, vecteurs_propres = np.linalg.eig(A)

print("Valeurs propres de A :", valeurs_propres)
print("Vecteurs propres de A :\n", vecteurs_propres)

# Construction de la matrice P (vecteurs propres) et D (
  matrice diagonale)
P = vecteurs_propres
D = np.diag(valeurs_propres)

# Calcul de P inverse
P_inv = np.linalg.inv(P)

# Vérification que A = P D P^{-1}
A_reconstruit = P @ D @ P_inv
print("A reconstruite (P D P^{-1}) :\n", A_reconstruit)
print("A est égale à P D P^{-1} :", np.allclose(A,
  A_reconstruit))
```

10.2 Interprétation géométrique

Lorsque A est diagonalisable, cela signifie que l'application linéaire associée à A agit de manière simple dans la base des vecteurs propres : chaque vecteur propre est simplement multiplié par sa valeur propre. Géométriquement, cela signifie que l'application linéaire étire ou contracte l'espace le long des directions des vecteurs propres, sans mélanger ces directions. Les axes définis par les vecteurs propres sont appelés les **axes propres** de la transformation.

Dans cette base, les effets de la matrice sont plus faciles à comprendre et à calculer, car la matrice est diagonale et les calculs sont simplifiés.

10.3 Conditions de diagonalisation

Une matrice A de taille $n \times n$ est diagonalisable si et seulement si elle admet une base de vecteurs propres, c'est-à-dire si l'espace vectoriel \mathbb{K}^n peut être décomposé en somme directe de ses sous-espaces propres. Cela se produit si et seulement si les conditions suivantes sont satisfaites :

- Le polynôme caractéristique de A est scindé sur \mathbb{K} , c'est-à-dire qu'il peut être factorisé en produit de polynômes de degré 1 à coefficients dans \mathbb{K} . Cela signifie que toutes les valeurs propres de A appartiennent à \mathbb{K} .
- Pour chaque valeur propre λ de A , la multiplicité géométrique (dimension du sous-espace propre E_λ) est égale à sa multiplicité algébrique (ordre de multiplicité de λ en tant que racine du polynôme caractéristique).

Lorsque ces conditions sont remplies, les vecteurs propres associés aux différentes valeurs propres forment une famille libre (linéairement indépendante), et l'on peut construire une base de \mathbb{K}^n composée de vecteurs propres de A .

10.4 Procédure de diagonalisation

Pour diagonaliser une matrice A , on suit les étapes suivantes :

1. **Calcul des valeurs propres** : Résoudre l'équation caractéristique $\det(A - \lambda I) = 0$ pour trouver les valeurs propres λ de A .
2. **Détermination des vecteurs propres** : Pour chaque valeur propre λ , trouver une base du sous-espace propre E_λ en résolvant le système linéaire $(A - \lambda I)\vec{v} = \vec{0}$.
3. **Vérification des conditions de diagonalisation** : Vérifier que la somme des dimensions des sous-espaces propres est égale à n , c'est-à-dire que les sous-espaces propres engendrent tout l'espace \mathbb{K}^n , et que les multiplicités géométriques sont égales aux multiplicités algébriques pour chaque valeur propre.
4. **Construction de la matrice P** : Former la matrice P dont les colonnes sont des vecteurs propres linéairement indépendants de A .

5. **Construction de la matrice diagonale D** : Construire la matrice D diagonale, dont les éléments diagonaux sont les valeurs propres de A , dans le même ordre que les vecteurs propres dans P .
6. **Vérification de la relation $A = PDP^{-1}$** : Confirmer que A peut être exprimée sous la forme $A = PDP^{-1}$.

10.5 Cas des matrices non diagonalisables

Toutes les matrices ne sont pas diagonalisables. Si une matrice ne satisfait pas les conditions de diagonalisation (par exemple, si la somme des dimensions des sous-espaces propres est strictement inférieure à n), alors elle n'est pas diagonalisable. Dans de tels cas, on peut envisager d'autres formes normales, comme la **trigonalisation** (réduction à une forme triangulaire supérieure), que nous aborderons dans la section suivante.

10.6 Exemple

Considérons la matrice :

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

Nous allons diagonaliser A .

Étape 1 : Calcul des valeurs propres

Le polynôme caractéristique de A est :

$$\chi_A(\lambda) = \det(A - \lambda I) = (1 - \lambda)(2 - \lambda)^2.$$

Les valeurs propres sont donc $\lambda = 1$ (multiplicité algébrique 1) et $\lambda = 2$ (multiplicité algébrique 2).

Étape 2 : Détermination des vecteurs propres

Pour $\lambda = 1$:

Réolvons $(A - I)\vec{v} = \vec{0}$:

$$(A - I) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Le système associé est :

$$\begin{cases} 1x_1 + 1x_2 = 0, \\ 1x_1 + 1x_3 = 0. \end{cases}$$

La première équation donne $x_1 = -x_2$, la seconde $x_1 = -x_3$. Donc $x_2 = x_3$.

Un vecteur propre associé est donc $\vec{v}_1 = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}$.

Pour $\lambda = 2$:

Réolvons $(A - 2I)\vec{v} = \vec{0}$:

$$(A - 2I) = \begin{pmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

Le système associé est :

$$\begin{cases} -1x_1 = 0, \\ x_1 = 0, \\ x_1 = 0. \end{cases}$$

On en déduit $x_1 = 0$, et x_2, x_3 sont libres. Donc les vecteurs propres associés sont de la forme $\vec{v} = \begin{pmatrix} 0 \\ x_2 \\ x_3 \end{pmatrix}$.

Une base du sous-espace propre E_2 est donnée par les vecteurs $\vec{v}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ et $\vec{v}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$.

Étape 3 : Vérification des conditions de diagonalisation

La multiplicité algébrique de $\lambda = 1$ est 1, sa multiplicité géométrique est aussi 1 (dimension de E_1).

La multiplicité algébrique de $\lambda = 2$ est 2, sa multiplicité géométrique est aussi 2 (dimension de E_2).

La somme des dimensions des sous-espaces propres est $1 + 2 = 3$, égale à n .

Donc A est diagonalisable.

Étape 4 : Construction de P et D

La matrice P est formée des vecteurs propres :

$$P = (\vec{v}_1 \ \vec{v}_2 \ \vec{v}_3) = \begin{pmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

La matrice diagonale D est :

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

Étape 5 : Vérification de $A = PDP^{-1}$

On peut vérifier que $A = PDP^{-1}$ en calculant PDP^{-1} et confirmant que cela donne A .

Interprétation

Dans la base formée des vecteurs propres de A , la matrice de A est diagonale. Cela signifie que l'application linéaire associée à A dilate ou contracte l'espace uniquement le long des directions des vecteurs propres, sans provoquer de rotation ou de mélange entre ces directions.

10.7 Remarques

- Si une matrice n'est pas diagonalisable, cela signifie qu'elle ne possède pas une base complète de vecteurs propres. Dans ce cas, on peut chercher à la mettre sous forme triangulaire supérieure (trigonalisation), ce qui est toujours possible sur un corps algébriquement clos. La trigonalisation sera abordée dans la section suivante.
- Une matrice diagonalisable est particulièrement simple à manipuler, notamment pour calculer ses puissances ou des fonctions de matrices (comme l'exponentielle de matrices), car les calculs se réduisent à des opérations sur les valeurs propres.
- La diagonalisation met en évidence la structure interne de l'application linéaire associée à la matrice, en identifiant les directions privilégiées (les axes propres) le long desquelles l'application agit de manière simple (étirement ou compression).
- En résumé, une matrice est diagonale dans la base de ses vecteurs propres.

```

import numpy as np

# Définition d'une matrice de Jordan non diagonalisable
A = np.array([[2, 1],
              [0, 2]])

# Calcul des valeurs propres et des vecteurs propres
valeurs_propres, vecteurs_propres = np.linalg.eig(A)

print("Valeurs propres de A :", valeurs_propres)
print("Vecteurs propres de A :\n", vecteurs_propres)

# Vérification si la matrice est diagonalisable
# Une matrice est diagonalisable si le rang de P est égal
  à la taille de la matrice
P = vecteurs_propres
rang_P = np.linalg.matrix_rank(P)
is_diagonalisable = (rang_P == A.shape[0])

print("Rang de la matrice de vecteurs propres P :",
      rang_P)
print("La matrice A est diagonalisable :",
      is_diagonalisable)

# Tentative de construction de P, D et  $P^{-1}$ 
try:
    D = np.diag(valeurs_propres)
    P_inv = np.linalg.inv(P)
    A_reconstruit = P @ D @ P_inv
    print("A reconstruite (P D  $P^{-1}$ ) :\n",
          A_reconstruit)
    print("A est égale à P D  $P^{-1}$  :", np.allclose(A,
          A_reconstruit))
except np.linalg.LinAlgError:
    print("La matrice P n'est pas inversible, donc A n'
    est pas diagonalisable.")

```

Dans le code ci-dessus on calcule le rang de la matrice qui contient les vecteurs propres, en effet il peut y avoir, lors du calcul avec numpy, un ou plusieurs vecteurs propres nuls (ce qui est impossible par définition) et donc on calcule le rang pour être sûr qu'on ne les compte pas.

11 Trigonalisation d'une matrice

11.1 Définition

La **trigonalisation** est un processus qui transforme une matrice carrée en une matrice triangulaire (supérieure ou inférieure) par un changement de base. Une matrice A est dite **trigonalisable** s'il existe une matrice inversible P telle que :

$$A = PTP^{-1},$$

où T est une matrice triangulaire supérieure (ou inférieure) et P est la matrice de passage correspondant au changement de base.

Contrairement à la diagonalisation, la trigonalisation est toujours possible sur un **corps algébriquement clos** (comme \mathbb{C}), même si la matrice A n'est pas diagonalisable. Dans T , les valeurs propres de A apparaissent sur la diagonale principale.

11.2 Conditions de trigonalisation

- Toute matrice carrée de taille $n \times n$ sur un corps **algébriquement clos** (comme \mathbb{C}) est trigonalisable. Cela résulte du théorème de Jordan et du fait que le polynôme caractéristique de A se scinde en facteurs linéaires.
- Sur un corps qui n'est pas algébriquement clos (par exemple, \mathbb{R}), la trigonalisation peut ne pas être possible si le polynôme caractéristique de A a des facteurs irréductibles de degré supérieur à 1. (Un facteur est dit **irréductible** sur un corps \mathbb{K} s'il ne peut pas être décomposé en produit de polynômes de degré strictement inférieur à lui-même avec des coefficients dans \mathbb{K} .)

11.3 Étapes du processus de trigonalisation

La trigonalisation consiste à trouver une base dans laquelle une matrice carrée A est représentée sous forme triangulaire supérieure. Voici un aperçu du processus:

1. **Calcul des valeurs propres:** Identifiez les valeurs propres de A en résolvant l'équation caractéristique $\det(A - \lambda I) = 0$.
2. **Détermination des vecteurs propres et généralisés :** Pour chaque valeur propre λ , trouvez une base du sous-espace propre $E_\lambda = \ker(A - \lambda I)$. Si la base n'est pas suffisante pour couvrir la dimension de l'espace, complétez-la avec des vecteurs généralisés en résolvant $(A - \lambda I)^k \vec{v} = 0$ pour $k > 1$.
3. **Formation de la matrice de passage P :** Construisez la matrice P en prenant les vecteurs propres et généralisés comme colonnes. Ces vecteurs doivent être linéairement indépendants.

4. **Calcul de la matrice triangulaire T** : Calculez $T = P^{-1}AP$, où T est triangulaire supérieure, et ses éléments diagonaux correspondent aux valeurs propres de A .

11.4 Méthodes avancées : Forme de Jordan

Pour des matrices de taille supérieure ou des cas plus complexes, des méthodes comme la **forme de Jordan** permettent de généraliser et de simplifier l'étude des matrices non diagonalisables. La forme de Jordan est une matrice presque diagonale composée de blocs triangulaires associés aux valeurs propres et à leurs vecteurs généralisés.

11.5 Lien avec la décomposition de Dunford

Pour les curieux, la **décomposition de Dunford** permet d'écrire une matrice A comme la somme de deux matrices $A = D + N$, où :

- D est diagonalisable.
- N est nilpotente ($N^k = 0$ pour un certain entier k).

Cette décomposition est particulièrement utile pour analyser les transformations linéaires. Elle généralise l'idée de trigonalisation et diagonalisation, en séparant clairement la partie "diagonalisable" de la partie "complexe" de la matrice.

12 Décomposition en valeurs singulières

Nous allons découvrir dans cette section la décomposition en valeurs singulières, une des techniques fondamentales de l'algèbre linéaire, utilisée pour résoudre une très grande quantité de problèmes : régression linéaire, réduction de dimension (PCA) et bien d'autres encore. La SVD (Singular Value Decomposition) est réellement une technique incontournable, utilisée en mécanique, électronique, physique, mathématique, et nos ingénieurs l'utilisent parfois sans le savoir pour construire nos ponts, nos avions, nos voitures, etc.

12.1 Introduction à la SVD

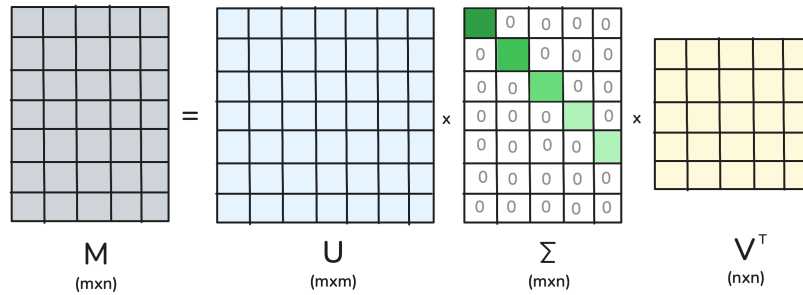
La décomposition en valeurs singulières (SVD) est une méthode de factorisation de matrice qui permet la décomposition en éléments propres de matrices de taille quelconque $n \times m$.

Le principe est de décomposer n'importe quelle matrice $M \in \mathcal{M}_{n,m}(\mathbb{K})$ tel que:

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

où :

- M est la matrice à décomposer (taille $m \times n$),
- U est une matrice orthogonale qui contient les vecteurs propres de la matrice MM^T (taille $m \times m$)
- Σ est une matrice diagonale (de la même taille que M) qui contient les valeurs singulières de M
- V est une matrice orthogonale qui contient les vecteurs propres de $M^T M$ (taille $n \times n$), donc V^T contient les vecteurs propres en ligne



12.2 Construction de matrices symétriques associées

Pour une matrice quelconque M , nous pouvons construire deux matrices symétriques positives semi-définies associées :

- $S_{\text{gauche}} = MM^T$, une matrice symétrique de taille $n \times n$.
- $S_{\text{droite}} = M^T M$, une matrice symétrique de taille $m \times m$.

Points importants :

1. Le produit d'une matrice rectangulaire M avec sa transposée MM^T est toujours une matrice symétrique, comme vu dans les chapitres précédents.
2. Les matrices S_{gauche} et S_{droite} sont positives semi-définies, ce qui signifie que toutes leurs valeurs propres sont non négatives.
3. Les vecteurs propres de S_{gauche} et S_{droite} correspondent respectivement aux vecteurs singuliers gauches et droits de M .

12.3 Calcul des valeurs et vecteurs singuliers

Les étapes pour calculer la SVD d'une matrice M sont les suivantes :

1. **Calcul des matrices symétriques associées :**

$$\begin{aligned} S_{\text{gauche}} &= MM^T, \\ S_{\text{droite}} &= M^T M. \end{aligned}$$

2. **Calcul des vecteurs propres et des valeurs propres :**

- Trouver les vecteurs propres $\{u_i\}$ et les valeurs propres $\{\lambda_i\}$ de S_{gauche} .
- Trouver les vecteurs propres $\{v_i\}$ et les mêmes valeurs propres $\{\lambda_i\}$ de S_{droite} .

3. **Construction des matrices U , Σ et V :**

- U est formée des vecteurs propres u_i normalisés de S_{gauche} .
- V est formée des vecteurs propres v_i normalisés de S_{droite} .
- Σ est une matrice diagonale dont les éléments diagonaux sont les racines carrées des valeurs propres $\sigma_i = \sqrt{\lambda_i}$, organisées en ordre décroissant.

Remarque importante : Les valeurs singulières de M sont les racines carrées des valeurs propres non nulles de $M^T M$ ou MM^T .

```

import numpy as np

# Définition de la matrice M
M = np.array([[1, 2],
              [3, 4],
              [5, 6]])

# Calcul de la SVD
U, Sigma, VT = np.linalg.svd(M)

print("Matrice U :\n", U)
print("Valeurs singulières :", Sigma)
print("Matrice V^T :\n", VT)

```

Interprétation des résultats :

- La matrice U contient les vecteurs singuliers gauches.
- Σ est un vecteur contenant les valeurs singulières de M .
- La matrice V^T contient les vecteurs singuliers droits transposés.

Pour reconstruire M à partir de U , Σ et V^T , nous devons étendre Σ en une matrice diagonale de la même taille que M .

```

# Extension de Sigma en une matrice diagonale de la
  taille de M
Sigma_full = np.zeros((M.shape[0], M.shape[1]))
np.fill_diagonal(Sigma_full, Sigma)

# Reconstruction de M
M_reconstructed = U @ Sigma_full @ VT

print("Matrice M reconstruite :\n", M_reconstructed)

```

Vérification :

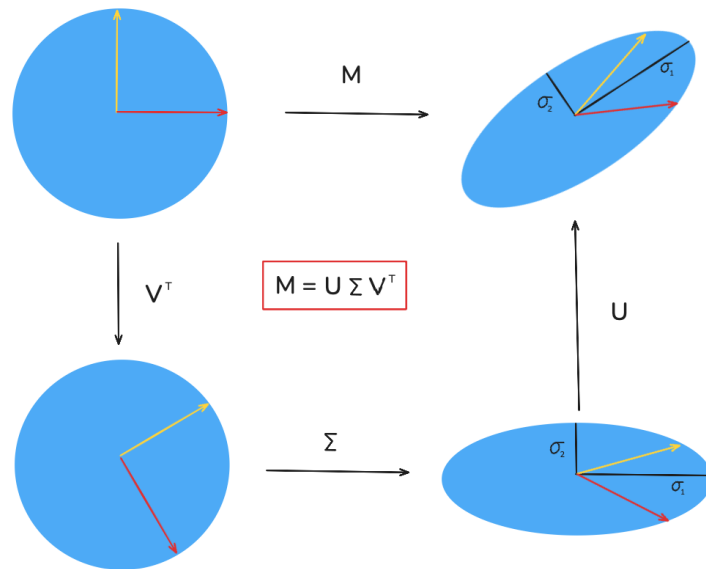
La matrice M reconstruite doit être égale à la matrice originale M (à des erreurs numériques près).

12.4 Visualisation géométrique de la SVD

La décomposition en valeurs singulières peut être interprétée géométriquement comme une transformation séquentielle de l'espace vectoriel. Elle décompose toute transformation linéaire, représentée par une matrice M , en trois étapes fondamentales :

1. **Rotation initiale** : La matrice V^T est une matrice orthogonale qui effectue une rotation de l'espace de départ. Elle aligne les vecteurs propres droits (ou vecteurs singuliers droits) de M avec les axes de l'espace.
2. **Mise à l'échelle** : La matrice diagonale Σ effectue une mise à l'échelle le long des axes transformés. Les valeurs singulières contenues dans Σ étirent ou contractent l'espace le long de ces axes. (Suivant la taille de la matrice, on peut avoir une diminution ou une augmentation de la dimension de l'espace, comme expliqué dans la section Matrices Rectangulaires).
3. **Rotation finale** : La matrice U est une autre matrice orthogonale qui effectue une rotation de l'espace après la mise à l'échelle. Elle aligne les axes avec les vecteurs propres gauches (ou vecteurs singuliers gauches) de M .

Ainsi, la SVD décompose une transformation linéaire complexe en rotations et mises à l'échelle simples, ce qui facilite grandement son interprétation et son analyse.



12.5 Approximation de rang faible et applications

Une des applications puissantes de la SVD est l'approximation de matrices par des matrices de rang inférieur. En ne conservant que les k plus grandes valeurs singulières et les vecteurs singuliers correspondants, nous pouvons obtenir une approximation de M qui minimise l'erreur au sens des moindres carrés.

$$M \approx U_k \Sigma_k V_k^T$$

où U_k , Σ_k et V_k sont obtenus en ne conservant que les k premiers termes. C'est à dire les k premières colonnes de U , les k plus grandes valeurs singulières et les k premières lignes de V^T

Exemple Numérique avec NumPy :

Supposons que nous souhaitons approximer la matrice M précédemment définie en utilisant seulement son premier vecteur singulier.

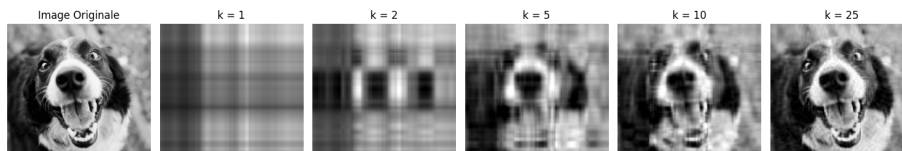
```
# Sélection du premier vecteur singulier
k = 1
U_k = U[:, :k]
Sigma_k = Sigma[:k]
VT_k = VT[:k, :]

# Reconstruction approximative de M
Sigma_k_matrix = np.diag(Sigma_k)
M_approx = U_k @ Sigma_k_matrix @ VT_k

print("Approximation de M avec rang k = 1 :\n", M_approx)
```

Application en compression d'images :

La SVD est largement utilisée pour la compression d'images. Une image en niveaux de gris peut être représentée comme une matrice dont les éléments correspondent aux intensités des pixels. En appliquant la SVD et en ne conservant qu'un nombre réduit de valeurs singulières, nous pouvons obtenir une version compressée de l'image. Par exemple:



Qui est produit avec le code suivant :

```
import matplotlib.pyplot as plt
from matplotlib.image import imread
import numpy as np

image = imread('image.jpg').astype(float)
k_values = [1, 2, 5, 10, 25]
compressed_images = []

for k in k_values:
    # Initialisation de la liste pour les canaux compressés
    channels = []

    # Application de la SVD sur chaque canal
    for i in range(3): # Pour les canaux Rouge, Vert, Bleu
        M = image[:, :, i]

        # Calcul de la SVD
        U, Sigma, VT = np.linalg.svd(M, full_matrices=False)

        # Approximation avec k valeurs singulières
        U_k = U[:, :k]
        Sigma_k = Sigma[:k]
        VT_k = VT[:k, :]

        M_approx = np.dot(U_k, np.dot(np.diag(Sigma_k), VT_k))
```



```

        channels.append(M_approx)

    # Reconstruction de l'image compressée en combinant
    les canaux
    compressed_image = np.stack(channels, axis=2)

    # S'assurer que les valeurs des pixels sont dans l'
    intervalle [0, 255]
    compressed_image = np.clip(compressed_image, 0, 255)

    compressed_images.append(compressed_image.astype('
uint8'))

plt.figure(figsize=(15, 5))

# Affichage de l'image originale
plt.subplot(1, len(k_values) + 1, 1)
plt.imshow(image.astype('uint8'))
plt.title('Image Originale')
plt.axis('off')

# Affichage des images compressées
for i, k in enumerate(k_values):
    plt.subplot(1, len(k_values) + 1, i + 2)
    plt.imshow(compressed_images[i])
    plt.title(f'k = {k}')
    plt.axis('off')

plt.tight_layout()
plt.show()

```

12.6 Lien entre la SVD et la Réduction de Dimension

La SVD est également au cœur de nombreuses techniques de réduction de dimension, dont l'Analyse en Composantes Principales (ACP). Sans entrer dans les détails de l'ACP, nous pouvons développer une intuition sur la façon dont la SVD facilite la réduction de dimension.

En appliquant la SVD à une matrice de données centrée X , nous obtenons :

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

Où :

- X est la matrice de données centrée (chaque ligne représente une observation, chaque colonne une variable).
- U contient les vecteurs singuliers gauches.
- Σ est une matrice diagonale des valeurs singulières.
- V^T contient les vecteurs singuliers droits transposés.

Intuition : Les valeurs singulières dans Σ indiquent l'importance de chaque composante dans la variance totale des données. En ne conservant que les k plus grandes valeurs singulières et les vecteurs singuliers correspondants, nous pouvons projeter les données dans un espace de dimension réduite qui capture l'essentiel de l'information.

Illustration avec NumPy :

Voici comment utiliser la SVD pour réduire la dimension des données tout en conservant la majeure partie de la variance.

```
import numpy as np

# Supposons que X est une matrice de données centrée (
    n_samples, n_features)
X = ... # Votre matrice de données centrée

# Calcul de la SVD
U, Sigma, VT = np.linalg.svd(X, full_matrices=False)

# Choix du nombre de dimensions k à conserver
k = 2 # Par exemple, réduire à 2 dimensions

# Construction de Sigma_k
Sigma_k = np.diag(Sigma[:k])

# Projection des données dans l'espace réduit
X_reduced = U[:, :k] @ Sigma_k

print("Données projetées dans un espace de dimension réduite :")
print(X_reduced)
```

Explications :

- Centrage des données : Avant d'appliquer la SVD, il est important de centrer les données en soustrayant la moyenne de chaque variable.
- Sélection des composantes principales : Les colonnes de U associées aux plus grandes valeurs singulières correspondent aux directions de plus grande variance dans les données.
- Projection : En multipliant U_k par Σ_k , nous obtenons les données projetées dans l'espace de dimension k .

Conclusion :

Cette approche permet de réduire la dimensionnalité des données tout en conservant l'information la plus significative. La SVD fournit ainsi un outil puissant pour simplifier des jeux de données complexes et faciliter leur analyse.

12.7 La SVD et le Pseudo-Inverse de Moore-Penrose

La décomposition en valeurs singulières (SVD) offre une approche élégante pour calculer l'inverse d'une matrice, ainsi que sa pseudo-inverse lorsque la matrice n'est pas inversible.

Inverse d'une matrice via la SVD

Supposons que la matrice M est carrée et inversible. Alors, sa SVD est donnée par $M = U\Sigma V^T$, et son inverse peut être exprimé comme :

$$M^{-1} = (U\Sigma V^T)^{-1} = V(\Sigma^{-1})U^T$$

Cette égalité est possible car les matrices U et V sont orthogonales, donc $U^{-1} = U^T$ et $V^{-1} = V^T$. De plus, la matrice Σ étant diagonale, son inverse est obtenue en inversant simplement chaque élément diagonal.

Problèmes d'instabilité numérique

Cependant, cette méthode peut être sujette à des instabilités numériques. En effet, les très petites valeurs singulières (par exemple, de l'ordre de 10^{-15}) peuvent refléter des erreurs de précision et deviennent extrêmement grandes lorsqu'on les inverse, ce qui peut conduire à des résultats erronés.

Calcul du Pseudo-Inverse de Moore-Penrose

Le pseudo-inverse de Moore-Penrose (notée M^+) est calculée de manière similaire à l'inverse complet, à la différence près que nous n'inversons que les éléments diagonaux **non nuls** de Σ au lieu de tenter d'inverser tous les éléments. En pratique, "non nuls" signifie "au-dessus d'un certain seuil" pour tenir compte des erreurs de précision.

Ainsi, le pseudo-inverse est donnée par :

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

où Σ^+ est obtenue en inversant les valeurs singulières non nulles de Σ et en laissant les autres éléments à zéro.

Avantages du Pseudo-Inverse

Cette approche est non seulement simple et intuitive, mais elle s'applique également aux matrices non carrées. En fait :

- Le pseudo-inverse de Moore-Penrose d'une matrice "tall" (plus de lignes que de colonnes) équivaut à son inverse à gauche.
- Le pseudo-inverse d'une matrice "wide" (plus de colonnes que de lignes) équivaut à son inverse à droite.

Voici comment calculer le pseudo-inverse de Moore-Penrose avec NumPy :

```
import numpy as np

# Définition de la matrice M (exemple non carré)
M = np.array([[1, 2, 3],
              [4, 5, 6]])

# Calcul de la SVD
U, Sigma, VT = np.linalg.svd(M, full_matrices=False)

# Inversion des valeurs singulières non nulles avec un
seuil
tol = 1e-15
Sigma_inv = np.array([1/s if s > tol else 0 for s in
                      Sigma])

# Construction de la matrice diagonale de la pseudo-
inverse de Sigma
Sigma_inv_matrix = np.diag(Sigma_inv)

# Calcul du pseudo-inverse de M
M_pseudo_inv = VT.T @ Sigma_inv_matrix @ U.T

print("Pseudo-inverse de M :\n", M_pseudo_inv)
```

Vérification

Le pseudo-inverse vérifie certaines propriétés, par exemple :

$$MM^+M = M \quad \text{et} \quad M^+MM^+ = M^+$$

Nous pouvons vérifier cela numériquement :

```
# Vérification de la propriété  $M M^+ M = M$ 
MMpM = M @ M_pseudo_inv @ M
print("M M^+ M :\n", MMpM)
print("Différence M M^+ M - M :\n", MMpM - M)

# Vérification de la propriété  $M^+ M M^+ = M^+$ 
MpMMp = M_pseudo_inv @ M @ M_pseudo_inv
print("M^+ M M^+ :\n", MpMMp)
print("Différence M^+ M M^+ - M^+ :\n", MpMMp -
      M_pseudo_inv)
```

12.8 Lien avec le chapitre sur le déterminant

Grâce à la décomposition en valeurs singulières, nous comprenons mieux le calcul du **pseudo-déterminant** d'une matrice, particulièrement lorsque cette matrice est singulière (non inversible). Le pseudo-déterminant est une généralisation du déterminant pour les matrices qui ne sont pas de rang plein, en ne considérant que les dimensions où la transformation n'écrase pas l'espace.

Pour une matrice A de rang r ayant r valeurs singulières non nulles $\sigma_1, \sigma_2, \dots, \sigma_r$, le pseudo-déterminant est donné par :

$$\text{pdet}(A) = \prod_{i=1}^r \sigma_i$$

Cette formule signifie que le pseudo-déterminant est le produit des valeurs singulières non nulles de A . Cela correspond au produit des facteurs d'étirement (ou de contraction) appliqués par A dans les directions où elle agit de manière non dégénérée.

Interprétation géométrique :

- Le déterminant d'une matrice carrée inversible représente le facteur de changement de volume induit par la transformation linéaire associée à cette matrice
- Lorsque la matrice est singulière (son déterminant classique est nul), cela indique que le volume est écrasé dans au moins une direction, et donc le déterminant ne fournit pas d'information utile sur les dimensions restantes
- Le pseudo-déterminant, en ne considérant que les valeurs singulières non nulles, mesure le changement de volume dans les sous-espaces où la transformation est inversible

Lien avec la SVD :

La SVD décompose A en $A = U\Sigma V^T$, où Σ est une matrice diagonale contenant les valeurs singulières σ_i . Les valeurs singulières nulles correspondent aux directions dans lesquelles A écrase l'espace (c'est-à-dire, réduit la dimension). En calculant le pseudo-déterminant, nous ignorons ces directions écrasées et nous concentrons sur les dimensions où l'action de A est significative.

12.9 Remarque avec NumPy

Lors de l'utilisation de la fonction `np.linalg.svd()` en Python pour calculer la décomposition en valeurs singulières, le paramètre `full_matrices` détermine la taille des matrices U et V^T retournées :

- Si `full_matrices=True` (valeur par défaut), les matrices U et V^T sont de taille complète. Cela signifie que même si la matrice originale M n'est pas carrée, U sera de taille $m \times m$ et V^T de taille $n \times n$, où M est de taille $m \times n$. Des zéros supplémentaires sont ajoutés pour compléter les matrices, ce qui peut entraîner des calculs inutiles et une utilisation excessive de la mémoire.
- Si `full_matrices=False`, les matrices U et V^T retournées sont de taille réduite, ne contenant que les composantes essentielles pour reconstruire M . Plus précisément, U sera de taille $m \times k$ et V^T de taille $k \times n$, où $k = \min(m, n)$.

Pourquoi utiliser `full_matrices=False` ?

Dans le contexte du calcul du pseudo-inverse de Moore-Penrose et des applications de la SVD, il est souvent plus efficace d'utiliser les matrices réduites pour plusieurs raisons :

- **Efficacité Computationnelle** : Les matrices réduites sont plus petites, ce qui réduit le temps de calcul et la consommation de mémoire, surtout pour les grandes matrices non carrées.
- **Focus sur les Valeurs Singulières Non Nulles** : En travaillant avec les matrices réduites, nous nous concentrons sur les valeurs singulières non nulles, qui sont les seules contribuant à la reconstruction de M et au calcul de sa pseudo-inverse.
- **Éviter les Zéros Superflus** : Les matrices complètes incluent des zéros supplémentaires qui n'apportent aucune information utile et peuvent compliquer les calculs.