Mathématiques pour l'Intelligence Artificielle

Licence d'Excellence

Partie 1 : Algèbre linéaire et Calcul différentiel pour l'IA

Par : PR. ZAKARY Omar

Faculté des Sciences Ben M'Sik, Université Hassan II de Casablanca

Sommaire

Introduction Générale

Cette partie du cours est conçue pour fournir aux étudiants une solide formation en mathématiques appliquées et en optimisation pour l'intelligence artificielle (IA). Il couvre des concepts clés issus de l'algèbre linéaire et du calcul différentiel, tous essentiels pour comprendre et appliquer des techniques modernes d'apprentissage automatique et d'IA.

Le but principal de ce cours est de donner aux étudiants les outils mathématiques nécessaires pour résoudre des problèmes complexes en intelligence artificielle. Les sujets abordés sont organisés en deux grandes parties :

- **Algèbre Linéaire pour l'IA** : L'algèbre linéaire est fondamentale dans la représentation et la manipulation des données. Nous couvrirons :
  - Les vecteurs, matrices et opérations matricielles
  - Les espaces vectoriels et sous-espaces
  - Les valeurs propres et vecteurs propres
  - La décomposition en valeurs singulières (SVD)

Ces concepts sont cruciaux pour la réduction de dimension, le traitement des données massives, et la classification en apprentissage automatique.

- **Calcul Différentiel pour l'IA** : Le calcul différentiel est essentiel pour l'optimisation et la modélisation en IA. Nous aborderons :
  - Les fonctions scalaires et vectorielles
  - Les dérivées partielles et gradients
  - Les intégrales de ligne et de surface
  - Les théorèmes fondamentaux (Green, Stokes, divergence)
  - Les méthodes d'optimisation et descente de gradient

Ces outils sont indispensables pour l'optimisation des modèles d'apprentissage automatique et la compréhension des algorithmes d'apprentissage profond.

Les mathématiques sont au cœur de l'intelligence artificielle. Qu'il s'agisse d'optimiser des modèles d'apprentissage, de réduire la dimensionnalité des données ou de comprendre des structures complexes, les techniques abordées dans ce cours fournissent une base solide pour construire des solutions performantes et robustes en IA. Par exemple :

- L'algèbre linéaire permet de manipuler efficacement les données multidimensionnelles et de découvrir des structures cachées dans les données.
- Le calcul différentiel fournit les outils nécessaires pour optimiser les modèles d'apprentissage et améliorer leurs performances.

Ce cours mettra l'accent sur l'application pratique des concepts mathématiques aux problèmes réels d'IA. Les étudiants auront l'occasion non seulement de comprendre les fondements théoriques, mais aussi de les appliquer concrètement à travers :

- Des exemples pratiques tirés de problèmes réels d'IA

3

— Des exercices de programmation et d'implémentation
— Des études de cas en apprentissage automatique
— Des projets d'application concrète

À la fin de ce cours, les étudiants seront capables de :

— Maîtriser les concepts fondamentaux de l'algèbre linéaire et du calcul différentiel appliqués à l'IA
— Comprendre et implémenter des algorithmes de réduction de dimensionnalité
— Optimiser des modèles d'apprentissage automatique
— Analyser et interpréter des résultats mathématiquement
— Appliquer ces connaissances à des problèmes concrets en IA et en science des données

Chapitre 1

Algèbre Linéaire pour l'IA

1.1 Introduction

1.1.1 L'importance de l'algèbre linéaire en IA

L'intelligence artificielle (IA) est fondée sur des algorithmes complexes qui manipulent de grandes quantités de données. Pour traiter ces données de manière efficace, l'algèbre linéaire joue un rôle central. Elle fournit les outils mathématiques nécessaires pour représenter et transformer les données sous forme de vecteurs et de matrices, facilitant ainsi la création de modèles capables d'apprendre et de faire des prédictions.

En IA, chaque élément de données (comme une image, un texte ou une entrée de capteur) est souvent converti en vecteur. Les opérations de base en algèbre linéaire telles que la multiplication matricielle, les transformations linéaires, et les décompositions matricielles sont cruciales pour le fonctionnement des réseaux de neurones, les méthodes de réduction de dimensionnalité (telles que PCA), et les algorithmes d'apprentissage supervisé et non supervisé

1.1.2 Pourquoi l'algèbre linéaire ?

En IA, les données sont souvent représentées sous forme de matrices (par exemple, des jeux de données avec des milliers de variables). Les opérations matricielles permettent d'effectuer des transformations complexes comme la rotation ou le redimensionnement des données.

Les modèles d'IA, en particulier les réseaux de neurones, s'appuient sur des transformations linéaires pour modéliser les relations entre les données d'entrée et de sortie. La multiplication de matrices permet de calculer les pondérations et les biais dans ces réseaux.

Les algorithmes d'optimisation, utilisés dans l'apprentissage automatique, reposent souvent sur des concepts d'algèbre linéaire comme les gradients, les matrices hessiennes, et la décomposition en valeurs singulières

5

(SVD), pour minimiser les erreurs et améliorer les performances des modèles.

1.1.3 Applications en IA

Réseaux de Neurones Artificiels (ANN) : Les couches d'un réseau de neurones peuvent être interprétées comme des transformations linéaires suivies de fonctions d'activation non-linéaires. Les poids et les biais sont stockés sous forme de matrices, et chaque propagation de données à travers le réseau implique des opérations matricielles.

Réduction de Dimensionnalité : Techniques telles que l'Analyse en Composantes Principales (PCA) utilisent des concepts d'algèbre linéaire pour projeter les données dans des espaces de dimension inférieure, tout en conservant l'essentiel de l'information.

Systèmes de Recommandation : La décomposition en valeurs singulières (SVD) est souvent utilisée dans les systèmes de recommandation pour réduire les matrices utilisateur-produit, facilitant ainsi des recommandations personnalisées basées sur des facteurs cachés.

L'algèbre linéaire constitue donc le fondement théorique indispensable pour aborder et maîtriser les algorithmes d'intelligence artificielle. Elle permet de comprendre les mécanismes internes de ces algorithmes, d'améliorer leur efficacité, et de développer des modèles performants pour résoudre des problèmes complexes du monde réel.

1.2 Vecteurs et Matrices

1.2.1 Vecteurs

Un **vecteur** est une entité mathématique qui a à la fois une magnitude (longueur) et une direction. Il est généralement représenté par une liste de nombres (ses coordonnées), chacun indiquant la position du vecteur dans un espace de dimension donnée.

Définition

Définition 1. Un vecteur en dimension n est une séquence ordonnée de n nombres réels, notée :

$$v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

où $v_1, v_2, ..., v_n$ sont les composantes du vecteur v.

Exemple 2. * En dimension 2 (plan), un vecteur pourrait être représenté comme : $v = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$

Ce vecteur a une direction et une longueur dans un plan à deux dimensions.

* En dimension 3 (espace), un vecteur pourrait être : $v = \begin{pmatrix} 1 \\ -2 \\ 5 \end{pmatrix}$

Ce vecteur a une composante sur chacun des trois axes : (Ox), (Oy) et (Oz).

Notations :

— **Vecteur colonne :** Un vecteur est souvent noté sous forme de colonne, comme dans les exemples ci-dessus.

— **Vecteur ligne :** Il peut également être écrit sous forme de ligne :

$$v = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \end{pmatrix}$$

Cette notation est souvent utilisée en analyse matricielle.

— **Représentation Géométrique :** Dans un espace à deux ou trois dimensions, un vecteur peut être représenté par une flèche partant de l'origine (point $(0,0)$ ou $(0,0,0)$) jusqu'au point déterminé par ses composantes.

1.2.2 Opérations sur les Vecteurs :

1.2.2.1 Addition de vecteurs :

Si $u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$ et $v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$ alors

$$u + v = \begin{pmatrix} u_1 + v_1 \\ u_2 + v_2 \\ \vdots \\ u_n + v_n \end{pmatrix}$$

Exemple : On considère les deux vecteurs $u = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ et $v = \begin{pmatrix} -2 \\ 3 \end{pmatrix}$ alors

$$u + v = \begin{pmatrix} 1-2 \\ 2+3 \end{pmatrix} = \begin{pmatrix} -1 \\ 5 \end{pmatrix}$$

1.2.2.2 Multiplication par un scalaire

Si $\alpha \in \mathbb{R}$ et $u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$ alors

$$\alpha u = \begin{pmatrix} \alpha u_1 \\ \alpha u_2 \\ \vdots \\ \alpha u_n \end{pmatrix}$$

Exemple : On considère le vecteur $u = \begin{pmatrix} -3\sqrt{3} \\ \frac{2}{3} \end{pmatrix}$ alors (pour $\alpha = \sqrt{2}$) on a :

$$\sqrt{2}u = \begin{pmatrix} \sqrt{2}\left(-3\sqrt{3}\right) \\ \sqrt{2}\left(\frac{2}{3}\right) \end{pmatrix} = \begin{pmatrix} -3\sqrt{6} \\ \frac{2\sqrt{2}}{3} \end{pmatrix}$$

1.2.2.3 Produit Scalaire

Le produit scalaire entre deux vecteurs $\mathbf{u}$ et $\mathbf{v}$ dans un espace vectoriel euclidien est défini comme suit. Si $\mathbf{u}$ et $\mathbf{v}$ sont des vecteurs dans $\mathbb{R}^n$, alors :

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i$$

où $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ et $\mathbf{v} = (v_1, v_2, \ldots, v_n)$. Le produit scalaire est également noté $\langle \mathbf{u}, \mathbf{v} \rangle$.
Le produit scalaire a plusieurs propriétés importantes :

- **Symétrie** : $\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u}$
- **Linéarité** : $\mathbf{u} \cdot (\mathbf{v} + \mathbf{w}) = \mathbf{u} \cdot \mathbf{v} + \mathbf{u} \cdot \mathbf{w}$
- **Positivité** : $\mathbf{u} \cdot \mathbf{u} \geq 0$, avec égalité si et seulement si $\mathbf{u} = \mathbf{0}$
- **Orthogonalité des vecteurs :** Deux vecteurs $\mathbf{u}$ et $\mathbf{v}$ sont orthogonaux si et seulement si $\mathbf{u} \cdot \mathbf{v} = 0$

1.2.2.4 Norme d'un Vecteur :

La norme d'un vecteur u, souvent notée $\|u\|$, représente la longueur ou la magnitude du vecteur. La norme $\|\mathbf{u}\|$ est définie par :

$$\|\mathbf{u}\| = \sqrt{\mathbf{u} \cdot \mathbf{u}} = \sqrt{\sum_{i=1}^{n} u_i^2}$$

Voici quelques types de normes courantes :

— **Norme Euclidienne** (ou norme L^2) :

$$\|\mathbf{u}\|_2 = \sqrt{\sum_{i=1}^{n} u_i^2}$$

— **Norme** L^1 (ou norme de Manhattan) :

$$\|\mathbf{u}\|_1 = \sum_{i=1}^{n} |u_i|$$

— **Norme** L^∞ (ou norme du maximum) :

$$\|\mathbf{u}\|_\infty = \max_i |u_i|$$

Exemple : On considère le vecteur $u = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$ alors on a :

$$\|u\| = \sqrt{(-1)^2 + (2)^2} = \sqrt{5}$$

Soit v un vecteur qui est égal au vecteur u normalisé, alors

$$v = \frac{1}{\sqrt{5}} \begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

Ainsi on a : $\|v\| = 1$

1.2.2.5 Produit Vectoriel

Le **produit vectoriel** de deux vecteurs $\mathbf{a}$ et $\mathbf{b}$ dans $\mathbb{R}^3$ est un vecteur $\mathbf{c} = \mathbf{a} \times \mathbf{b}$ qui est perpendiculaire aux deux vecteurs $\mathbf{a}$ et $\mathbf{b}$, et dont la norme est donnée par :

$$\|\mathbf{a} \times \mathbf{b}\| = \|\mathbf{a}\| \|\mathbf{b}\| \, |\sin\theta|$$

où θ est l'angle entre $\mathbf{a}$ et $\mathbf{b}$.

Le produit vectoriel peut être calculé à l'aide du déterminant suivant, pour deux vecteurs $\mathbf{a} = (a_1, a_2, a_3)$ et $\mathbf{b} = (b_1, b_2, b_3)$:

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = \mathbf{i}(a_2 b_3 - a_3 b_2) - \mathbf{j}(a_1 b_3 - a_3 b_1) + \mathbf{k}(a_1 b_2 - a_2 b_1)$$

où $\mathbf{i}$, $\mathbf{j}$, et $\mathbf{k}$ sont les vecteurs unitaires dans les directions x, y et z, respectivement.

Exemple : Prenons $\mathbf{a} = (2, 0, -1)$ et $\mathbf{b} = (1, -1, 3)$. Calculons le produit vectoriel $\mathbf{a} \times \mathbf{b}$.

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 2 & 0 & -1 \\ 1 & -1 & 3 \end{vmatrix} = \mathbf{i}(0 \cdot 3 - (-1) \cdot (-1)) - \mathbf{j}(2 \cdot 3 - (-1) \cdot 1) + \mathbf{k}(2 \cdot (-1) - 0 \cdot 1)$$
$$= -\mathbf{i} - 7\mathbf{j} - 2\mathbf{k} = (-1, -7, -2)$$

1.2.2.6 En Python

Une liste est une structure de données native en Python qui peut être utilisée pour représenter un vecteur.

```
# Définir un vecteur en tant que liste Python
vecteur = [1, 2, 3, 4]
```

La bibliothèque **NumPy** est largement utilisée pour les opérations mathématiques en Python. Elle permet de définir des vecteurs sous forme de tableaux multidimensionnels appelés arrays.

```
import numpy as np
# Définir un vecteur avec NumPy
vecteur = np.array([1, 2, 3, 4])
```

NumPy est optimisé pour effectuer des calculs vectoriels et matriciels. Il permet de faire des opérations mathématiques comme la somme, le produit scalaire, ou l'addition de vecteurs de manière plus naturelle et efficace.

```
import numpy as np
# Définir un vecteur
vecteur = np.array([1, 2, 3, 4])

# Opérations
print(vecteur + 1)              # Ajouter 1 à chaque élément
print(vecteur * 2)              # Multiplier chaque élément par 2
print(np.dot(vecteur, vecteur))  # Produit scalaire de deux vecteurs
```

En **NumPy**, la calcul d'une norme peut être fait avec la fonction **np.linalg.norm**. La normalisation d'un vecteur consiste à diviser chaque élément du vecteur par sa norme pour obtenir un vecteur de norme 1

```
import numpy as np
# Définir un vecteur
vecteur = np.array([3, 4])

# Calculer la norme
norme = np.linalg.norm(vecteur)

# Normaliser le vecteur
vecteur_normalise = vecteur / norme

print("Vecteur normalisé :", vecteur_normalise)
```

1.2.3 Matrices

En algèbre linéaire, une matrice est une structure rectangulaire composée de nombres organisés en lignes et colonnes. Les matrices jouent un rôle central dans les algorithmes d'apprentissage automatique, notamment pour la manipulation de grandes quantités de données, la transformation des vecteurs, et les calculs dans les réseaux de neurones.

Les matrices sont essentielles pour de nombreuses opérations dans les algorithmes d'apprentissage automatique, notamment pour :

— **La manipulation des données :** une base de données contenant m exemples et n caractéristiques peut être représentée par une matrice de dimension $m \times n$.
— **Les réseaux de neurones :** les poids des connexions entre les neurones peuvent être représentés sous forme de matrices.
— **La réduction de dimension :** des techniques comme l'analyse en composantes principales (**PCA**) utilisent la multiplication de matrices pour réduire le nombre de caractéristiques tout en conservant un maximum d'information.

1.2.3.1 Définitions

Définition

Définition 3. Une matrice A de dimension $m \times n$ (avec m lignes et n colonnes) est notée de la manière suivante :

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_2n \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Chaque élément a_{ij} représente l'entrée à l'intersection de la i-ème ligne et de la j-ème colonne. Si $m = n$ la matrice est dite une matrice carrée.

Définition

Définition 4. La **transposée** d'une matrice A, notée A^T, est obtenue en échangeant ses lignes et ses colonnes.
Soit A une matrice de taille $m \times n$ définie par :

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

La transposée de A, notée A^T, est une matrice de taille $n \times m$ définie par :

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

Si $A = A^T$ alors A est dite symétrique.

Définition

Définition 5. Matrice identité : La matrice identité I_n (ou I si on a pas d'information sur la dimension) est une matrice carrée de dimension $n \times n$ où tous les éléments de la diagonale principale sont égaux à 1 et tous les autres éléments sont égaux à 0 :

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

Pour toute matrice A de dimension $n \times n$, nous avons : $A.I_n = A$ et $I_n.A = A$

1.2.3.2 Opérations sur les matrices

Addition de matrices : Si deux matrices A et B ont les mêmes dimensions $m \times n$, leur addition se fait en additionnant les éléments correspondants :

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

Multiplication d'une matrice par un scalaire : La multiplication d'une matrice A par un scalaire α consiste à multiplier chaque élément de A par α :

$$\alpha A = \begin{pmatrix} \alpha a_{11} & \alpha a_{12} & \cdots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \cdots & \alpha a_2 n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha a_{m1} & \alpha a_{m2} & \cdots & \alpha a_{mn} \end{pmatrix}$$

Multiplication de matrices : Si A est une matrice de dimension $m \times n$ et B une matrice de dimension $n \times p$, leur produit est une matrice C de dimension $m \times p$ définie par : $C = A \times B$

Chaque élément c_{ij} est donné par le produit scalaire entre la i-ème ligne de A et la j-ème colonne de B :

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

Ce produit n'est possible que si le nombre de colonnes de A est égal au nombre de lignes de B.

Exemple 6. Soit A une matrice de dimension 2×3 et B une matrice de dimension 3×2 définies par :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix}$$

Le produit $C = A \cdot B$ est donné par :

$$C = \begin{bmatrix} (1 \cdot 7 + 2 \cdot 9 + 3 \cdot 11) & (1 \cdot 8 + 2 \cdot 10 + 3 \cdot 12) \\ (4 \cdot 7 + 5 \cdot 9 + 6 \cdot 11) & (4 \cdot 8 + 5 \cdot 10 + 6 \cdot 12) \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

Le produit des matrice n'est pas commutatif.

Définition

Définition 7. Inverse d'une matrice : Si une matrice A est carrée et inversible, alors il existe une matrice A^{-1} telle que :

$$A.A^{-1} = A^{-1}.A = I_n$$

Proposition 8. *Une matrice carrée A est inversible si et seulement si le déterminant de A est non nul.*

L'inversion de matrices est une opération cruciale dans la résolution des systèmes d'équations linéaires et dans plusieurs algorithmes d'optimisation.

Définition

Définition 9. Matrice Orthogonale : Une matrice carrée $A \in \mathbb{R}^{n \times n}$ est dite **orthogonale** si ses colonnes (ou lignes) sont orthogonales entre elles et ont une norme unitaire. En d'autres termes, une matrice A est orthogonale si elle satisfait la condition suivante :

$$A^T A = A A^T = I$$

où A^T est la matrice transposée de A et I est la matrice identité de taille $n \times n$.

Exemple 10. 1) Considérons la matrice suivante : $A = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$

Pour vérifier si A est orthogonale, nous calculons $A^T A$: on a

$$A^T = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

Donc

$$A^T A = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$$

Ainsi, A est une matrice orthogonale.

2) Considérons la matrice suivante : $B = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$

Calculons $B^T B$: on a

$$B^T = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

$$B^T B = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$$

Ainsi, B est également une matrice orthogonale.

1.2.3.3 Valeurs et Vecteurs Propres Propres

Définition

Définition 11. Une **valeur propre** λ de la matrice A est un scalaire tel qu'il existe un vecteur non nul v (appelé **vecteur propre**) pour lequel l'équation suivante est vérifiée :

$$Av = \lambda v$$

Autrement dit, v est un vecteur propre associé à la valeur propre λ si v est un vecteur non nul qui, lorsqu'il est multiplié par la matrice A, est simplement multiplié par λ.

Les valeurs propres sont les racines du polynôme caractéristique de A, défini par :

$$p(\lambda) = \det(A - \lambda I)$$

où « det » est le déterminant et I est la matrice identité.

Pour trouver les vecteurs propres associés à une valeur propre donnée, on résout le système homogène $(A - \lambda I)v = 0$.

Exemple 12. Considérons la matrice suivante : $A = \begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix}$

Pour trouver les valeurs propres de A, on détermine le polynôme caractéristique :

$$\det(A - \lambda I) = \det \begin{pmatrix} 4 - \lambda & 1 \\ 2 & 3 - \lambda \end{pmatrix} = (4 - \lambda)(3 - \lambda) - 2$$

$$= \lambda^2 - 7\lambda + 10$$

Les valeurs propres sont les racines de ce polynôme, soit $\lambda_1 = 5$ et $\lambda_2 = 2$.

Pour chaque valeur propre, nous trouvons les vecteurs propres associés en résolvant le système $(A - \lambda I)v = 0$.

Trouvons les vecteurs propres
* Pour $\lambda_1 = 5$: On a

$$A - 5I = \begin{pmatrix} 4-5 & 1 \\ 2 & 3-5 \end{pmatrix} = \begin{pmatrix} -1 & 1 \\ 2 & -2 \end{pmatrix}$$

Nous résolvons le système $(A - 5I)v = 0$: En posant $v = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ on aura :

$$\begin{pmatrix} -1 & 1 \\ 2 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Le système équivalent est :

$$\begin{cases} -x_1 + x_2 = 0 \\ 2x_1 - 2x_2 = 0 \end{cases}$$

Le système équivalent est : $x_1 = x_2$. Donc on a $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_1 \end{pmatrix} = x_1 \begin{pmatrix} 1 \\ 1 \end{pmatrix}$
Un vecteur propre associé à $\lambda_1 = 5$ est donc :

$$v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

* Pour $\lambda_2 = 2$:

$$A - 2I = \begin{pmatrix} 4-2 & 1 \\ 2 & 3-2 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 2 & 1 \end{pmatrix}$$

Nous résolvons le système $(A - 2I)v = 0$, en posant $v = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ on aura :

$$\begin{pmatrix} 2 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Le système équivalent est : $2x_1 + x_2 = 0$. Ce qui simplifie à : $x_2 = -2x_1$
Un vecteur propre associé à $\lambda_2 = 2$ est donc :

$$v_2 = \begin{pmatrix} 1 \\ -2 \end{pmatrix}$$

Définition

Définition 13. Une matrice A **symétrique** est dite **définie positive** si elle est positive et inversible, autrement dit si elle vérifie l'une des quatre propriétés équivalentes suivantes :

1. Pour toute matrice colonne non nulle $\mathbf{x}$ à n éléments réels, on a : $\mathbf{x}^\mathsf{T} A\mathbf{x} > 0$. Autrement dit, la forme quadratique définie par A est strictement positive pour $\mathbf{x} \neq 0$.
2. Toutes les valeurs propres de A (qui sont nécessairement réelles) sont strictement positives.
3. La forme bilinéaire symétrique $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad (\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}^\mathsf{T} A\mathbf{y}$ est un produit scalaire sur $\mathbb{R}^n$.
4. Il existe une matrice $M \in \mathcal{M}_n(\mathbb{R})$ inversible telle que $A = M^\mathsf{T} M$.

Exemple 14. Soit la matrice suivante :

$$C = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Soit le vecteur $\mathbf{v} = \begin{pmatrix} x \\ y \end{pmatrix}$. Calculons le produit scalaire $\mathbf{v}^T C \mathbf{v}$.

Le produit scalaire est donné par :

$$\mathbf{v}^T C \mathbf{v} = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = x^2 - y^2.$$

Puisque $\mathbf{v}^T C \mathbf{v}$ peut être positif ou négatif en fonction de x et y, la matrice C n'est ni définie positive ni définie négative

Exemple 15. Soit la matrice suivante :

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

Calculons les mineurs principaux de la matrice A.

Les mineurs principaux de A sont :

$$H_1 = A_{11} = 4 \quad \text{et} \quad \det(A) = \det \begin{pmatrix} 4 & 2 \\ 2 & 3 \end{pmatrix} = 12 - 4 = 8.$$

Puisque $H_1 > 0$ et $\det(A) > 0$, la matrice A est définie positive.

1.2.3.4 En Python

En Python, vous pouvez définir une matrice en utilisant plusieurs méthodes, mais la bibliothèque **NumPy** est la plus couramment utilisée pour manipuler des matrices efficacement.

```
import numpy as np
# Définir une matrice 2x3
matrice = np.array([[1, 2, 3],
                    [4, 5, 6]])
print(matrice)
```

NumPy offre des fonctions pratiques pour créer des matrices spéciales comme des matrices de zéros, de uns, ou une matrice identité.

```
import numpy as np

# Matrice de 3x3 remplie de zéros
matrice_zeros = np.zeros((3, 3))
print(matrice_zeros)

# Matrice de 2x4 remplie de uns
matrice_uns = np.ones((2, 4))
print(matrice_uns)

# Matrice identité 3x3
matrice_identite = np.eye(3)
print(matrice_identite)
```

Une fois une matrice créée avec **NumPy**, vous pouvez facilement manipuler ses éléments, accéder aux lignes/colonnes spécifiques, ou encore faire des opérations mathématiques.

```python
import numpy as np

# Définir une matrice 2x3
matrice = np.array([[1, 2],
                    [4, 5]])

# Accéder à l'élément de la première ligne, deuxième colonne
element = matrice[0, 1]
print("Élément (1ère ligne, 2ème colonne) :", element)

# Accéder à la première colonne
colonne = matrice[:, 0]
print("Première colonne :", colonne)

# Accéder à la deuxième ligne
ligne = matrice[1, :]
print("Deuxième ligne :", ligne)

# Calculer la transposée
transposee = np.transpose(matrice)

# Autre méthode pour obtenir la transposée
transposee = matrice.T

# Calculer le déterminant
determinant = np.linalg.det(matrice)

# Calculer l'inverse
inverse = np.linalg.inv(matrice)

# Vérification : Le produit de A et A^{-1}
identite = np.dot(matrice, inverse)
```

En Python, pour calculer les valeurs propres et les vecteurs propres d'une matrice, la bibliothèque **NumPy** propose une fonction appelée **np.linalg.eig**. Cette fonction prend une matrice carrée comme entrée et renvoie deux éléments : Les valeurs propres et Les vecteurs propres.

```python
import numpy as np
# Définir une matrice carrée 2x2
matrice = np.array([[4, 2],
                    [1, 3]])

# Calculer les valeurs propres et les vecteurs propres
valeurs_propres, vecteurs_propres = np.linalg.eig(matrice)

# Afficher les valeurs propres
print("Valeurs propres :", valeurs_propres)

# Afficher les vecteurs propres
print("Vecteurs propres :")
print(vecteurs_propres)
```

1.3 Décomposition en Valeurs Singulières (SVD)

La décomposition en valeurs singulières (SVD, pour **Singular Value Decomposition**) est un outil mathématique fondamental en intelligence artificielle (IA) et en apprentissage automatique. Elle est utilisée dans diverses applications pour analyser, simplifier ou décomposer les données.

La SVD est une factorisation d'une matrice A en trois matrices particulières. Si A est une matrice de dimension $m \times n$, alors il existe une décomposition de la forme :

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

où :

— $U \in \mathbb{R}^{m\times m}$ est une matrice orthogonale dont les colonnes sont les vecteurs propres normalisés de AA^T,

— $\Sigma \in \mathbb{R}^{m\times n}$ est une matrice diagonale avec les valeurs singulières de A sur la diagonale. Les valeurs singulières sont les racines carrées des valeurs propres de A^TA,

— $V \in \mathbb{R}^{n\times n}$ est une matrice orthogonale dont les colonnes sont les vecteurs propres normalisés de A^TA.

Plus précisément :

$$U = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \quad V = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}$$

où $\sigma_1, \sigma_2, \ldots, \sigma_r$ sont les valeurs singulières de A, et r est le rang de A. Les vecteurs u_i et v_i sont les vecteurs propres associés aux valeurs propres de AA^T et A^TA, respectivement.

Une convention courante est de ranger les valeurs $\Sigma_{i,i}$ par ordre décroissant. Alors, la matrice Σ est déterminée de façon unique par A (mais U et V ne le sont pas). La SVD est utile pour plusieurs applications :

— **Réduction de dimension** : En utilisant les k premières valeurs singulières et les colonnes correspondantes des matrices U et V, on peut obtenir une approximation de rang k de la matrice A.

— **Compression** : La SVD permet de compresser des données tout en préservant leur structure principale.

— **Dénombrement de données** : Les valeurs singulières peuvent aider à identifier les dimensions principales dans un espace de données.

Exemple 16. 1) Considérons la matrice A suivante :

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

Pour cette matrice diagonale, les matrices U et V seront les matrices identité de taille 3×3 et Σ sera identique à A.

1. Calcul de U : Pour une matrice diagonale, U est la matrice identité.

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

2. Calcul de Σ : C'est la matrice contenant les valeurs singulières sur la diagonale.

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

3. Calcul de V : Également, pour une matrice diagonale, V est la matrice identité.

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

Ainsi, la décomposition en valeurs singulières de A est :

$$A = U\Sigma V^T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}^T$$

Ce qui donne :

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

*** Remarque :** Par convention on doit ranger les valeurs $\Sigma_{i,i}$ par ordre décroissant, alors on obtient la décomposition suivante

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

Ce qui donne

$$U = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \text{ et } V = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Ainsi $A = U\Sigma V^T = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}^T$

2) Considérons la matrice suivante :

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

Nous allons calculer la décomposition en valeurs singulières (SVD) de A, c'est-à-dire exprimer A sous la forme :

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

Étape 1 : Calcul des valeurs propres de A^TA

Tout d'abord, calculons A^TA (le produit de la transposée de A et de A) :

$$A^TA = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$$

Ensuite, déterminons les valeurs propres de A^TA en résolvant le déterminant de $A^TA - \lambda I$:

$$\det(A^TA - \lambda I) = \det\left(\begin{bmatrix} 5-\lambda & 4 \\ 4 & 5-\lambda \end{bmatrix}\right) = (5-\lambda)^2 - 16 = \lambda^2 - 10\lambda + 9$$

Les racines de cette équation quadratique sont les valeurs propres de A^TA :

$$\lambda_1 = 9, \quad \lambda_2 = 1$$

Les valeurs singulières de A sont les racines carrées des valeurs propres de A^TA :

$$\sigma_1 = \sqrt{9} = 3, \quad \sigma_2 = \sqrt{1} = 1$$

Étape 2 : Calcul des vecteurs propres de A^TA

Les vecteurs propres associés aux valeurs propres $\lambda_1 = 9$ et $\lambda_2 = 1$ peuvent être obtenus en résolvant $(A^TA - \lambda I)v = 0$.

Pour $\lambda_1 = 9$:

$$(A^TA - 9I)v = \begin{bmatrix} -4 & 4 \\ 4 & -4 \end{bmatrix} v = 0$$

Cela donne le vecteur propre $v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Pour $\lambda_2 = 1$:

$$(A^TA - I)v = \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix} v = 0$$

Cela donne le vecteur propre $v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$.

Étape 3 : Construction des matrices U, Σ et V

La matrice V est formée par les vecteurs propres normalisés de A^TA :

$$V = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

La matrice Σ est une matrice diagonale avec les valeurs singulières σ_1 et σ_2 :

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

Enfin, la matrice U peut être calculée comme $U = AV\Sigma^{-1}$, ou en utilisant les vecteurs propres normalisés de AA^T.

Après calcul, on obtient :

$$U = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

Conclusion

La décomposition en valeurs singulières de la matrice A est donc :

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^T$$

Cette décomposition montre comment une matrice A peut être exprimée sous forme de matrices orthogonales et diagonales, ce qui est utile pour la compression, la réduction de dimension, et d'autres applications en algèbre linéaire.

1.4 Espaces vectoriels et sous-espaces

Définition

Définition 17. Un **espace vectoriel** est un ensemble de vecteurs, associés à deux opérations : l'addition de vecteurs et la multiplication par un scalaire, satisfaisant certaines propriétés. Plus formellement, soit V un ensemble de vecteurs sur un corps $\mathbb{K}$ (typiquement $\mathbb{R}$ ou $\mathbb{C}$), alors V est un espace vectoriel si pour tout $\mathbf{u}, \mathbf{v} \in V$ et pour tout $a, b \in \mathbb{K}$, les conditions suivantes sont vérifiées :
Addition de vecteurs : L'addition de deux vecteurs est encore un vecteur dans V, i.e. $\mathbf{u} + \mathbf{v} \in V$.
Multiplication par un scalaire : La multiplication d'un vecteur par un scalaire reste dans V, i.e. $a\mathbf{u} \in V$.
Propriétés de l'addition :
Commutativité : $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$.

Associativité : $(\mathbf{u}+\mathbf{v})+\mathbf{w}=\mathbf{u}+(\mathbf{v}+\mathbf{w})$.
Existence d'un élément neutre : Il existe un vecteur nul $\mathbf{0}\in V$ tel que $\mathbf{u}+\mathbf{0}=\mathbf{u}$ pour tout $\mathbf{u}\in V$.
Existence d'un élément opposé : Pour chaque vecteur $\mathbf{u}\in V$, il existe un vecteur opposé $-\mathbf{u}\in V$ tel que $\mathbf{u}+-\mathbf{u}=\mathbf{0}$.
Propriétés de la multiplication par un scalaire :
Distributivité par rapport à l'addition de vecteurs : $a(\mathbf{u}+\mathbf{v})=a\mathbf{u}+a\mathbf{v}$.
Distributivité par rapport à l'addition de scalaires : $(a+b)\mathbf{u}=a\mathbf{u}+b\mathbf{u}$.
Associativité de la multiplication scalaire : $a(b\mathbf{u})=(ab)\mathbf{u}$.
Existence d'un élément neutre pour la multiplication scalaire : $1\mathbf{u}=\mathbf{u}$ pour tout $\mathbf{u}\in V$.

Définition

Définition 18. Un **sous-espace vectoriel** W d'un espace vectoriel V est un sous-ensemble de V tel que W est lui-même un espace vectoriel. Cela signifie que pour tout $\mathbf{u},\mathbf{v}\in W$ et pour tout $a\in\mathbb{K}$, on a :
$\mathbf{u}+\mathbf{v}\in W$ (fermeture par addition),
$a\mathbf{u}\in W$ (fermeture par multiplication scalaire),
Le vecteur nul de V est également dans W.

Un exemple important est celui de l'ensemble des solutions d'un système d'équations linéaires homogène, qui forme un sous-espace vectoriel appelé le **noyau** ou **espace nul** de la matrice associée au système. Un autre exemple est l'**image** d'une matrice, qui est le sous-espace engendré par les colonnes de la matrice.

Exemple 19. (Sous-espace vectoriel)
Considérons l'ensemble V des vecteurs de la forme

$$V=\left\{\begin{pmatrix}x_1\\x_2\end{pmatrix}\in\mathbb{R}^2\;\middle|\;x_1+x_2=0\right\}.$$

Nous allons vérifier les trois propriétés ci-dessous.

Contient le vecteur nul

Le vecteur nul $\mathbf{0}=\begin{pmatrix}0\\0\end{pmatrix}$ appartient à V puisque $0+0=0$.

Ainsi, l'ensemble V contient le vecteur nul.

Fermeture par addition : Soient $\mathbf{u}=\begin{pmatrix}u_1\\u_2\end{pmatrix}$ et $\mathbf{v}=\begin{pmatrix}v_1\\v_2\end{pmatrix}$ deux éléments de V, par définition on a $u_1+u_2=0$ et $v_1+v_2=0$,

On pose $\mathbf{w}=\mathbf{u}+\mathbf{v}=\begin{pmatrix}u_1+v_1\\u_2+v_2\end{pmatrix}=\begin{pmatrix}w_1\\w_2\end{pmatrix}$. On a $\mathbf{w}\in V\iff w_1+w_2=0$

Alors :

$$w_1+w_2=(u_1+v_1)+(u_2+v_2)=(u_1+u_2)+(v_1+v_2)=0+0=0.$$

Ainsi, $\mathbf{w}\in V$, donc V est fermé par addition.

Fermeture par multiplication scalaire :

Soit $\lambda\in\mathbb{R}$ et $\lambda\mathbf{u}=\lambda\begin{pmatrix}u_1\\u_2\end{pmatrix}=\begin{pmatrix}\lambda u_1\\\lambda u_2\end{pmatrix}$.

Alors :

$$\lambda u_1 + \lambda u_2 = \lambda(u_1 + u_2) = \lambda \times 0 = 0.$$

Ainsi, $\lambda \mathbf{u} \in V$, donc V est fermé par multiplication scalaire.

Conclusion : Puisque V satisfait les trois conditions (contient le vecteur nul, fermé par addition et multiplication scalaire), nous pouvons conclure que V est un sous-espace vectoriel de $\mathbb{R}^2$.

(Noyau d'une matrice)

Considérons le système d'équations linéaires homogène suivant :

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

Ce système peut être représenté sous forme matricielle :

$$A\mathbf{x} = \mathbf{0}, \quad \text{où} \quad A = \begin{pmatrix} 1 & 2 & -1 \\ 2 & 4 & -2 \\ -1 & -2 & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Le système $A\mathbf{x} = \mathbf{0}$ est homogène, donc nous cherchons l'ensemble des solutions qui forment le **noyau** (ou espace nul) de la matrice A. À partir de la première ligne, nous obtenons :

$$x_1 + 2x_2 - x_3 = 0 \quad \Rightarrow \quad x_1 = -2x_2 + x_3.$$

Alors nous avons la solution générale :

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -2x_2 + x_3 \\ x_2 \\ x_3 \end{pmatrix} = x_2 \begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix} + x_3 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

L'ensemble des solutions est donc un sous-espace vectoriel de dimension 2 engendré par les vecteurs :

$$\mathbf{v}_1 = \begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{v}_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

Ainsi, le **noyau** de la matrice A est donné par :

$$\ker(A) = \text{Vect}\,(\mathbf{v}_1, \mathbf{v}_2)\,.$$

Exemple 20. (Image d'une matrice)

Considérons la matrice A suivante :

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

L'image de la matrice A, notée $\text{Im}(A)$, est le sous-espace vectoriel engendré par les colonnes de A. Autrement dit, $\text{Im}(A)$ est l'ensemble des combinaisons linéaires des colonnes de A.

Les colonnes de A sont les vecteurs suivants :

$$\mathbf{c}_1 = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, \quad \mathbf{c}_2 = \begin{pmatrix} 2 \\ 4 \\ 2 \end{pmatrix}, \quad \mathbf{c}_3 = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}.$$

Nous remarquons que $\mathbf{c}_2$ et $\mathbf{c}_3$ sont des multiples de $\mathbf{c}_1$:

$$\mathbf{c}_2 = 2\mathbf{c}_1, \quad \mathbf{c}_3 = \mathbf{c}_1.$$

Par conséquent, les colonnes $\mathbf{c}_2$ et $\mathbf{c}_3$ ne sont pas linéairement indépendantes. L'image de la matrice A est donc engendrée par le seul vecteur $\mathbf{c}_1$.

Alors : L'image de A est un sous-espace vectoriel de $\mathbb{R}^3$ de dimension 1, et une base de cet espace est donnée par $\mathbf{c}_1$. Par conséquent :

$$\text{Im}(A) = \text{Vect}\left(\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \right).$$

1.5 Applications en IA :

1.5.1 Exemples

En apprentissage automatique, les données brutes sont souvent transformées en vecteurs afin de les manipuler plus facilement dans les algorithmes. Deux exemples clés de cette représentation sont l'analyse d'images et le traitement du langage naturel.

Représentation des images en tant que vecteurs

Prenons l'exemple d'une image en niveaux de gris de dimension 28 × 28 pixels. Chaque pixel de cette image a une intensité de gris entre 0 (noir) et 255 (blanc). Une image de 28 × 28 peut être représentée par une matrice de 28 lignes et 28 colonnes, où chaque élément de la matrice correspond à l'intensité d'un pixel.

Cependant, pour passer cette image à un algorithme d'apprentissage automatique, cette matrice doit être "aplatie" en un vecteur de dimension $28 \times 28 = 784$. Ce vecteur contiendra les intensités de chaque pixel sous la forme :

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{728} \end{pmatrix}$$

où x_1, x_2, ..., x_{728} représentent les intensités des pixels de l'image.

Cela permet de traiter l'image comme un point dans un espace vectoriel de dimension 784. Cette technique est largement utilisée dans les réseaux de neurones pour la classification d'images, notamment dans des applications comme MNIST, où chaque chiffre manuscrit est représenté comme un vecteur de 784 dimensions.

Représentation des mots en tant que vecteurs dans le traitement du langage naturel (NLP)

Dans le traitement automatique du langage naturel (NLP), les mots sont souvent représentés sous forme de vecteurs dans un espace de grande dimension. Cette technique permet de capturer des relations sémantiques entre les mots. L'une des approches les plus populaires est Word2Vec, qui transforme les mots en vecteurs denses de dimension d, où d est typiquement de l'ordre de quelques centaines (par exemple, $d = 300$).

Si nous considérons un vocabulaire de taille V, chaque mot est représenté par un vecteur $w_i \in \mathbb{R}^d$ pour $i = 1, 2, ..., V$.

La méthode **Word2Vec** apprend ces vecteurs à partir de grands corpus de texte, en s'assurant que les mots ayant des contextes similaires dans le corpus aient des vecteurs proches dans l'espace vectoriel.

Ces vecteurs sont souvent appelés **embeddings** et sont obtenus par des techniques d'apprentissage profond. La principale idée est de minimiser une fonction de coût qui garantit que les mots contextuellement proches sont également proches dans l'espace des vecteurs. Une des fonctions de coût classiques est la fonction de perte par **softmax** ou **negative sampling**.

Par exemple, si w_i et w_j sont deux mots proches dans une phrase, alors leurs vecteurs doivent maximiser la probabilité :

$$P\left(w_j|w_i\right) = \frac{e^{w_i.w_j}}{\sum_{k=1}^{V} e^{w_i.w_k}}$$

où $w_i.w_j$ est le produit scalaire entre les vecteurs w_i et w_j.

Cette technique permet de capturer des relations linguistiques telles que les analogies. Par exemple, après l'entraînement, on pourrait observer une relation comme :

vecteur("roi") − vecteur("homme") + vecteur("femme") ≈ vecteur("reine")

l'expression **− vecteur("homme")** ne représente pas exactement "pas un homme", mais elle capture une sorte de différence conceptuelle ou de "transformation" dans l'espace sémantique des vecteurs.

Dans cet espace de vecteurs, la proximité des vecteurs reflète les similarités sémantiques entre les mots. Par exemple, des mots ayant des significations proches ou souvent utilisés dans des contextes similaires (comme "roi" et "reine") auront des vecteurs proches.

Le terme **− vecteur("homme")** peut être interprété comme la soustraction des caractéristiques liées au concept de "homme" du vecteur "roi". Ainsi, on retire l'aspect masculin du vecteur "roi", et en ajoutant **vecteur("femme")**, on introduit les caractéristiques féminines. Le résultat de cette opération vectorielle est un vecteur proche de **vecteur("reine")**, ce qui illustre le changement du genre tout en conservant les caractéristiques royales.

Cela montre comment la représentation vectorielle des mots permet d'exprimer des relations sémantiques de manière numérique, utile pour des tâches telles que la traduction automatique, la génération de texte, et l'analyse de sentiment.

1.5.2 Détection des anomalies

La détection d'anomalies est une technique essentielle dans le domaine de l'apprentissage automatique et de l'analyse des données. L'une des approches pour détecter les anomalies dans un ensemble de données consiste à utiliser la décomposition en valeurs singulières (SVD). Dans cet exemple, nous allons démontrer comment la SVD peut être utilisée pour identifier des anomalies dans un ensemble de données.

1.5 Ensemble de Données

Considérons un ensemble de données sous forme de matrice A, où chaque ligne représente un échantillon et chaque colonne représente une caractéristique :

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

Dans cet ensemble de données, l'observation $(1, 1, 1)$ est suspectée d'être une anomalie.

1.5 Décomposition en Valeurs Singulières

Nous allons effectuer une décomposition en valeurs singulières de la matrice A. La SVD décompose la matrice A comme suit :

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

La décomposition consiste à écrire la matrice A sous la forme : $A = U \sum V^T$ avec U et V sont des matrices orthogonales et $\sum$ est une matrice diagonale, telles que U est de dimension 4×4, et $\sum$ est de dimension 4×3 et V est de dimension 3×3.

Étape 1 :

Puisque le nombre de lignes est supérieur au nombre de colonnes dans la matrice A, alors on commence par le calcul de U, pour ce faire on pose $B = AA^T$, donc on a

$$\begin{aligned} B &= AA^T \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 3 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{aligned}$$

Déterminons les valeurs propres et les vecteurs propres de B, alors on a

$$\begin{aligned}\det(B-\lambda I) &= \det\begin{pmatrix} 1-\lambda & 0 & 1 & 0 \\ 0 & 1-\lambda & 1 & 0 \\ 1 & 1 & 3-\lambda & 1 \\ 0 & 0 & 1 & 1-\lambda \end{pmatrix} \\ &= \lambda^4 - 6\lambda^3 + 9\lambda^2 - 4\lambda \\ &= \lambda\left(\lambda^3 - 6\lambda^2 + 9\lambda - 4\right) \\ &= \lambda(\lambda-1)^2(\lambda-4)\end{aligned}$$

Donc les valeurs propres de B sont 4 et 1 et 1 et 0.
Alors on a

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

Étape 2 :
Déterminons les vecteurs propres de B :
* Pour $\lambda_1 = 4$ on a

$$B - 4I = \begin{pmatrix} -3 & 0 & 1 & 0 \\ 0 & -3 & 1 & 0 \\ 1 & 1 & -1 & 1 \\ 0 & 0 & 1 & -3 \end{pmatrix}$$

Nous résolvons le système $(B-4I)u = 0$: En posant $u = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$ on aura :

$$\begin{pmatrix} -3 & 0 & 1 & 0 \\ 0 & -3 & 1 & 0 \\ 1 & 1 & -1 & 1 \\ 0 & 0 & 1 & -3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Le système équivalent est :

$$\begin{cases} x_3 - 3x_1 & = 0 \\ x_3 - 3x_2 & = 0 \\ x_1 + x_2 - x_3 + x_4 & = 0 \\ x_3 - 3x_4 & = 0 \end{cases}$$

Ce qui simplifie à : $x_3 = 3x_1$ et $x_3 = 3x_2$ et $x_3 = 3x_4$, donc $x_4 = x_2 = x_1$. Un vecteur propre associé à $\lambda_1 = 4$ est donc :

$$u = \begin{pmatrix} 1 \\ 1 \\ 3 \\ 1 \end{pmatrix}$$

Après normalisation on trouve

$$u_1 = \frac{1}{\sqrt{12}} \begin{pmatrix} 1 \\ 1 \\ 3 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{12}}{12} \\ \frac{\sqrt{12}}{12} \\ \frac{3\sqrt{12}}{12} \\ \frac{\sqrt{12}}{12} \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{6} \end{pmatrix}$$

* Pour $\lambda_2 = 1$ on a

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

Nous résolvons le système $(B - I)v = 0$: En posant $u = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$ on aura :

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Le système équivalent est :

$$\begin{cases} x_3 & = 0 \\ x_1 + x_2 + 2\,x_3 + x_4 & = 0 \end{cases}$$

Ce qui simplifie à : $x_3 = 0$ et $x_1 = -x_2 - x_4$. Donc on a $u = \begin{pmatrix} -x_2 - x_4 \\ x_2 \\ 0 \\ x_4 \end{pmatrix} = \begin{pmatrix} -x_2 \\ x_2 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ x_4 \end{pmatrix} =$

$x_2 \begin{pmatrix} -1 \\ 1 \\ 0 \\ 0 \end{pmatrix} + x_4 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

Alors les vecteurs propres associés à $\lambda_2 = 1$ sont donc :

$$u = \begin{pmatrix} -1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \text{ et } u' = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Après normalisation on trouve

$$u_2 = \begin{pmatrix} \frac{-\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \\ 0 \end{pmatrix} \text{ et } u_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

* Pour $\lambda_3 = 0$ on a

$$B - 0I = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 3 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Nous résolvons le système $(B - 0I)v = 0$: En posant $u = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$ on aura :

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 3 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Le système équivalent est :

$$\begin{cases} x_1 + x_3 & = 0 \\ x_2 + x_3 & = 0 \\ x_1 + x_2 + 3\,x_3 + x_4 & = 0 \\ x_3 + x_4 & = 0 \end{cases}$$

Ce qui simplifie à : $x_3 = -x_1$ et $x_3 = -x_2$ et $x_3 = -x_4$, donc $x_4 = x_2 = x_1$. Un vecteur propre associé à $\lambda_3 = 0$ est donc :

$$u = \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \end{pmatrix}$$

Après normalisation on trouve

$$u_4 = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

Finalement on a $U = (u_1, u_2, u_3, u_4)$ c'est à dire

$$U = \begin{pmatrix} \frac{\sqrt{3}}{6} & \frac{-\sqrt{2}}{2} & 0 & \frac{1}{2} \\ \frac{\sqrt{3}}{6} & \frac{\sqrt{2}}{2} & 0 & \frac{1}{2} \\ \frac{\sqrt{3}}{2} & 0 & 0 & -\frac{1}{2} \\ \frac{\sqrt{3}}{6} & 0 & 1 & \frac{1}{2} \end{pmatrix}$$

Étape 3 :

Pour calculer la matrice $V = (v_1, v_2, v_3)$ on peut calculer les vecteurs propres de $A^T A$, pour simplifier le calcul on peut calculer les vecteurs de V en utilisant la démarche suivante : On a

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

Ce qui donne $v_i = \frac{1}{\sigma_i} A^T.u_i$, donc on a :

$$v_1 = \frac{1}{\sigma_1} \times A^T.u_1$$

$$v_1 = \frac{1}{2} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} . \begin{pmatrix} \frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{6} \end{pmatrix}$$

$$v_1 = \begin{pmatrix} \frac{\sqrt{3}}{3} \\ \frac{\sqrt{3}}{3} \\ \frac{\sqrt{3}}{3} \end{pmatrix}$$

Et on a

$$v_2 = \frac{1}{\sigma_2} \times A^T.u_2$$

$$v_2 = \frac{1}{1} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} . \begin{pmatrix} \frac{\sqrt{2}}{2} \\ 0 \\ 0 \\ -\frac{\sqrt{2}}{2} \end{pmatrix}$$

$$v_2 = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ 0 \\ -\frac{\sqrt{2}}{2} \end{pmatrix}$$

Et on a

$$v_3 = \frac{1}{\sigma_3} \times A^T.u_3$$

$$v_3 = \frac{1}{1} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Donc la matrice V est donnée par

$$V = \begin{pmatrix} \frac{\sqrt{3}}{3} & \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{3}}{3} & 0 & 0 \\ \frac{\sqrt{3}}{3} & -\frac{\sqrt{2}}{2} & 1 \end{pmatrix}$$

Conclusion : Alors la décomposition en valeur singulière de A est

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

$$A = \begin{pmatrix} \frac{\sqrt{3}}{6} & \frac{-\sqrt{2}}{2} & 0 & \frac{1}{2} \\ \frac{\sqrt{3}}{6} & \frac{\sqrt{2}}{2} & 0 & \frac{1}{2} \\ \frac{\sqrt{3}}{2} & 0 & 0 & -\frac{1}{2} \\ \frac{\sqrt{3}}{6} & 0 & 1 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\sqrt{3}}{3} & \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{3}}{3} & 0 & 0 \\ \frac{\sqrt{3}}{3} & -\frac{\sqrt{2}}{2} & 1 \end{pmatrix}^T$$

1.5 Identification des Anomalies

Une fois que nous avons les matrices U, Σ, et V, nous pouvons reconstruire une approximation de la matrice A en utilisant un nombre réduit de valeurs singulières. Supposons que nous gardons seulement la première valeur singulière :

$$\begin{aligned} A_{approx} &= U_{[:,1]}\Sigma_{[1,1]}V^T_{[:,1]} \\ &= \begin{pmatrix} \frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{6} \\ \frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{6} \end{pmatrix} .2. \begin{pmatrix} \frac{\sqrt{3}}{3} \\ \frac{\sqrt{3}}{3} \\ \frac{\sqrt{3}}{3} \end{pmatrix}^T \\ &= \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & 1 & 1 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} \end{aligned}$$

Nous pouvons alors comparer la matrice A_{approx} avec A. Les points qui présentent des différences significatives entre ces deux matrices sont considérés comme des anomalies.

En ne conservant que la première valeur singulière, on a montré que la troisième ligne de la matrice A est beaucoup plus dominante que les autres lignes, ce qui la désigne comme une anomalie. Ce type de comportement est souvent détecté dans des scénarios de détection d'anomalies où une ou plusieurs observations se démarquent par leur influence disproportionnée sur les données.

Chapitre 2

Calcul Différentiel pour l'IA

2.1 Fonctions scalaires et vectorielles

2.1.1 Fonctions scalaires

Une fonction scalaire f est une fonction qui prend un ou plusieurs arguments et retourne un seul nombre réel. Par exemple, une fonction scalaire en une variable est de la forme :

$$f : \mathbb{R}^n \to \mathbb{R}, \quad x \mapsto f(x)$$

où $f(x)$ est un nombre réel pour chaque $x \in \mathbb{R}^n$.

Exemple 21. 1) Considérons la fonction scalaire suivante :

$$f(x) = 2x^2 + 3x - 5$$

Pour $x = 2$, nous avons :

$$f(2) = 2(2)^2 + 3(2) - 5 = 8 + 6 - 5 = 9$$

2) Considérons la fonction scalaire suivante :

$$f(x, y) = 2x^2 + 3y$$

Pour $x = 1$ et $y = 2$, nous avons :

$$f(1, 2) = 2 + 3 \times 2 = 8$$

En Python
Nous pouvons définir la fonction scalaire et évaluer sa valeur en $(1, 2)$ dans Python avec le code suivant :

```python
import sympy as sp
import numpy as np

# Définir les variables symboliques
x_sym, y_sym = sp.symbols('x y')

# Définir la fonction f(x, y)
f = 2*x_sym**2 + 3*y_sym

x_val=1
y_val=2

print(f"Pour: x = {x_val}, y = {y_val}, f(x, y) = {f.evalf(subs={x_sym: x_val, y_sym: y_val})}")
```

2.1.2 Fonctions vectorielles

Une fonction vectorielle prend un ou plusieurs arguments et retourne un vecteur. Par exemple, une fonction vectorielle en une variable est de la forme :

$$\mathbf{f} : \mathbb{R} \to \mathbb{R}^n, \quad t \mapsto \mathbf{f}(t)$$

où $\mathbf{f}(t)$ est un vecteur dans $\mathbb{R}^n$ pour chaque $t \in \mathbb{R}$.

Exemple 22. Considérons la fonction vectorielle suivante :

$$\mathbf{f}(t) = \begin{pmatrix} t^2 \\ 2t \\ 1 \end{pmatrix}$$

Pour $t = 1$, nous avons :

$$\mathbf{f}(1) = \begin{pmatrix} 1^2 \\ 2 \cdot 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

2.2 Dérivées partielles et gradients

2.2.1 Dérivées partielles

La dérivée partielle d'une fonction $f(x_1, x_2, \ldots, x_n)$ par rapport à la variable x_i est donnée par :

$$\frac{\partial f}{\partial x_i} = \lim_{\Delta x_i \to 0} \frac{f(x_1, \ldots, x_i + \Delta x_i, \ldots, x_n) - f(x_1, \ldots, x_i, \ldots, x_n)}{\Delta x_i}$$

Cela mesure comment la fonction change lorsque x_i change, tandis que toutes les autres variables sont maintenues constantes.

Exemple 23. Considérons la fonction $f(x, y) = x^2y + 3xy^2$. Les dérivées partielles sont :

$$\frac{\partial f}{\partial x} = 2xy + 3y^2$$

$$\frac{\partial f}{\partial y} = x^2 + 6xy$$

Pour $x = 1$ et $y = 2$, nous avons :

$$\left.\frac{\partial f}{\partial x}\right|_{(1,2)} = 2 \cdot 1 \cdot 2 + 3 \cdot 2^2 = 4 + 12 = 16$$

$$\left.\frac{\partial f}{\partial y}\right|_{(1,2)} = 1^2 + 6 \cdot 1 \cdot 2 = 1 + 12 = 13$$

2.2.2 Gradient

Le gradient d'une fonction scalaire $f(x_1, x_2, \ldots, x_n)$ est un vecteur de ses dérivées partielles :

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right)$$

Il indique la direction du plus grand taux de variation de la fonction.

Exemple 24. Considérons la fonction scalaire $f(x, y) = x^2 + y^2$. Le gradient est :

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x \\ 2y \end{pmatrix}$$

Pour $x = 1$ et $y = 2$, nous avons :

$$\left.\nabla f\right|_{(1,2)} = \begin{pmatrix} 2 \cdot 1 \\ 2 \cdot 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$$

2.2.3 Divergence

La divergence d'un champ vectoriel $\mathbf{F} = (F_1, F_2, F_3)$ dans l'espace à trois dimensions est une fonction scalaire qui mesure la variation de la quantité sortante du champ vectoriel à un point donné. Elle est définie par :

$$\operatorname{div} \mathbf{F} = \nabla \cdot \mathbf{F} = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} + \frac{\partial F_3}{\partial z}$$

où $\nabla \cdot \mathbf{F}$ est l'opérateur divergence, et F_1, F_2, F_3 sont les composantes du champ vectoriel $\mathbf{F}$ selon les directions x, y, et z respectivement.

Exemple 25. Soit le champ vectoriel $\mathbf{F} = (x, y, z)$, qui correspond à un champ de position en trois dimensions. Calculons sa divergence :

$$\mathbf{F}(x, y, z) = (x, y, z)$$

La divergence de $\mathbf{F}$ est donnée par :

$$\nabla \cdot \mathbf{F} = \frac{\partial x}{\partial x} + \frac{\partial y}{\partial y} + \frac{\partial z}{\partial z}$$

Calculons chaque dérivée partielle :

$$\frac{\partial x}{\partial x} = 1, \quad \frac{\partial y}{\partial y} = 1, \quad \frac{\partial z}{\partial z} = 1$$

Donc, la divergence de $\mathbf{F}$ est :

$$\nabla \cdot \mathbf{F} = 1 + 1 + 1 = 3$$

La divergence du champ vectoriel $\mathbf{F}(x, y, z) = (x, y, z)$ est donc constante et égale à 3.

2.2.4 En Python

Nous pouvons calculer le gradient (les dérivées partielles) en Python avec l'approche suivante :

```python
import sympy as sp
import numpy as np

# Définir les variables symboliques
x_sym, y_sym = sp.symbols('x y')

# Définir la fonction f(x, y)
f = x_sym**2*y_sym + 3*x_sym*y_sym**2

# Calcul des dérivées partielles (gradient)
df_dx = sp.diff(f, x_sym)
df_dy = sp.diff(f, y_sym)

# Convertir les dérivées en fonctions lambdas pour évaluation numérique
grad_f_x = sp.lambdify([x_sym, y_sym], df_dx, 'numpy')
grad_f_y = sp.lambdify([x_sym, y_sym], df_dy, 'numpy')

x_val=1
y_val=2
print(f"Pour: x = {x_val}, y = {y_val}, df/dx = {grad_f_x(x_val, y_val)} et df/dy = {
    grad_f_y(x_val, y_val)}")
```

2.3 Optimisation

2.3.1 Points critiques

Les points critiques d'une fonction f sont les points où le gradient est nul :

$$\nabla f = \mathbf{0}$$

Ces points sont candidats pour être des extrémums locaux.

Exemple 26. Considérons la fonction $f(x, y) = x^2 + y^2 - 4x - 6y + 13$. Le gradient est :

$$\nabla f = \begin{pmatrix} 2x - 4 \\ 2y - 6 \end{pmatrix}$$

Les points critiques satisfont $\nabla f = \mathbf{0}$:

$$2x - 4 = 0 \implies x = 2$$

$$2y - 6 = 0 \implies y = 3$$

Donc, le point critique est $(2, 3)$.

2.3.2 Extrémums

Les points critiques peuvent être des minimums locaux, maximums locaux ou des points selle. Pour déterminer le type de point critique, on utilise souvent le *test de la dérivée seconde* ou la *matrice Hessienne* :

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

- Si la matrice Hessienne est définie positive (Toutes les valeurs propres de H sont strictement positives), le point est un minimum local.
- Si elle est définie négative (Toutes les valeurs propres de H sont strictement négatives), le point est un maximum local.
- Si elle n'est ni positive ni négative définie, le point est un point selle.

Exemple 27. (Minimum local) Dans l'exemple précédent on a trouvé comme point critique $(2, 3)$. Pour vérifier le type de ce point critique, calculons la matrice Hessienne de la fonction :

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

La matrice Hessienne est définie positive, donc le point critique $(2, 3)$ est un minimum local.

Exemple 28. (Maximum local) Considérons la fonction suivante :

$$f(x, y) = -x^2 - y^2 + 4$$

Cette fonction est une paraboloïde inversée.

Trouvons les points critiques : Le gradient est :

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} -2x \\ -2y \end{pmatrix}$$

Pour trouver les points critiques, nous résolvons :

$$\nabla f = \mathbf{0} \implies \begin{pmatrix} -2x \\ -2y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Cela donne :

$$x = 0 \quad \text{et} \quad y = 0$$

Donc, le point critique est $(0, 0)$.

Vérification du type de point critique : Calculons la matrice Hessienne :

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}$$

Les valeurs propres de H sont -2 et -2, qui sont toutes négatives. La matrice Hessienne est donc définie négative, et le point $(0, 0)$ est un **maximum local**.

Exemple 29. (Point selle) Considérons la fonction suivante :

$$g(x, y) = x^2 - y^2$$

Cette fonction est une hyperboloïde à une nappe.

Trouvons les points critiques : Le gradient est

$$\nabla g = \begin{pmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x \\ -2y \end{pmatrix}$$

Pour trouver les points critiques, nous résolvons :

$$\nabla g = \mathbf{0} \implies \begin{pmatrix} 2x \\ -2y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Cela donne :

$$x = 0 \quad \text{et} \quad y = 0$$

Donc, le point critique est $(0, 0)$.

Vérification du type de point critique : Calculons la matrice Hessienne

$$H = \begin{bmatrix} \frac{\partial^2 g}{\partial x^2} & \frac{\partial^2 g}{\partial x \partial y} \\ \frac{\partial^2 g}{\partial y \partial x} & \frac{\partial^2 g}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

Les valeurs propres de H sont 2 et -2. La matrice Hessienne a des valeurs propres de signes opposés, ce qui signifie qu'elle est indéfinie. Le point $(0, 0)$ est donc un point selle.

2.3.3 Application en IA

Dans le cadre de l'apprentissage automatique, on cherche à minimiser une fonction de coût $J(\theta_1, \theta_2)$ en fonction de deux variables θ_1 et θ_2. La fonction de coût est donnée par :

$$J(\theta_1, \theta_2) = (\theta_1 - 2)^2 + 2(\theta_2 - 3)^2.$$

1. Trouver les points critiques de la fonction de coût en calculant le gradient et en résolvant $\nabla J(\theta_1, \theta_2) = 0$.
2. Calculer la matrice hessienne H de la fonction de coût.
3. Déterminer si le point critique trouvé est un minimum, un maximum ou un point selle à l'aide de la matrice hessienne.

Solution :

Étape 1 : Calcul du gradient et des points critiques La fonction de coût est :

$$J(\theta_1, \theta_2) = (\theta_1 - 2)^2 + 2(\theta_2 - 3)^2.$$

Le gradient est le vecteur des dérivées partielles :

$$\nabla J(\theta_1, \theta_2) = \begin{pmatrix} \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \end{pmatrix} = \begin{pmatrix} 2(\theta_1 - 2) \\ 4(\theta_2 - 3) \end{pmatrix}.$$

Pour trouver les points critiques, on résout :

$$\begin{pmatrix} 2(\theta_1 - 2) \\ 4(\theta_2 - 3) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Ainsi, on obtient les points critiques :

$$\theta_1 = 2 \quad \text{et} \quad \theta_2 = 3.$$

Le point critique est donc $(\theta_1, \theta_2) = (2, 3)$.

Étape 2 : Calcul de la matrice hessienne La matrice hessienne H est la matrice des dérivées secondes de la fonction de coût. Calculons chaque dérivée seconde :

$$\frac{\partial^2 J}{\partial \theta_1^2} = 2, \quad \frac{\partial^2 J}{\partial \theta_1 \partial \theta_2} = 0, \quad \frac{\partial^2 J}{\partial \theta_2 \partial \theta_1} = 0, \quad \frac{\partial^2 J}{\partial \theta_2^2} = 4.$$

La matrice hessienne est donc :

$$H = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}.$$

Étape 3 : Analyse de la matrice hessienne Pour déterminer la nature du point critique, nous analysons les valeurs propres de la matrice hessienne. Puisque la matrice hessienne est diagonale, ses valeurs propres sont ses éléments diagonaux, c'est-à-dire 2 et 4.

Comme les deux valeurs propres sont strictement positives, la matrice hessienne est définie positive. Cela implique que le point critique $(\theta_1, \theta_2) = (2, 3)$ est un minimum local.

Cela signifie que l'algorithme d'optimisation va converger vers ce point pour minimiser la fonction $J(\theta_1, \theta_2)$, ce qui peut correspondre à la meilleure solution pour un problème d'apprentissage automatique.

2.3.4 En Python

Nous pouvons calculer le point critique, la matrice Hessienne et ses valeurs propres en Python avec l'approche suivante :

```python
import sympy as sp
# Définir les variables symboliques
x, y = sp.symbols('x y')

# Définir la fonction
f(x, y) f = x**2 + y**2 - 4*x - 6*y + 13

# Calculer les dérivées partielles (gradient)
df_dx = sp.diff(f, x)
df_dy = sp.diff(f, y)

# Résoudre le système d'équations pour trouver les points critiques (où le gradient est nul)
solutions = sp.solve([df_dx, df_dy], (x, y))

# Afficher les points critiques
print("Points critiques:", solutions)

# Calculer la matrice Hessienne
Hessian = sp.hessian(f, (x, y))

# Afficher la matrice Hessienne
print("\nMatrice Hessienne:") sp.pprint(Hessian)

# Calculer les valeurs propres de la matrice Hessienne
eigenvalues = Hessian.eigenvals()

# Afficher les valeurs propres
print("\nValeurs propres de la matrice Hessienne:", eigenvalues)
```

2.3.5 Descente de gradient

La méthode de la descente de gradient est un algorithme pour trouver les minimums locaux d'une fonction. On met à jour les paramètres en suivant la direction opposée au gradient :

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)$$

où α est le taux d'apprentissage, et $\mathbf{x}_k$ est la position actuelle. Cette procédure est répétée jusqu'à convergence.

Exemple 30. Considérons la fonction $f(x, y) = x^2 + y^2$ avec un taux d'apprentissage $\alpha = 0.1$. Si nous partons du point $\mathbf{x}_0 = (2, 2)$, nous mettons à jour les paramètres comme suit :

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)$$

Le gradient de cette fonction est :

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x \\ 2y \end{pmatrix}$$

Initialisation :

$$\mathbf{x}_0 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

Calcul de $\mathbf{x}_1$:

$$\nabla f(\mathbf{x}_0) = \begin{pmatrix} 2 \cdot 2 \\ 2 \cdot 2 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix}$$

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0) = \begin{pmatrix} 2 \\ 2 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 4 \\ 4 \end{pmatrix} = \begin{pmatrix} 1.6 \\ 1.6 \end{pmatrix}$$

Calcul de $\mathbf{x}_2$:

$$\nabla f(\mathbf{x}_1) = \begin{pmatrix} 2 \cdot 1.6 \\ 2 \cdot 1.6 \end{pmatrix} = \begin{pmatrix} 3.2 \\ 3.2 \end{pmatrix}$$

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha \nabla f(\mathbf{x}_1) = \begin{pmatrix} 1.6 \\ 1.6 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 3.2 \\ 3.2 \end{pmatrix} = \begin{pmatrix} 1.28 \\ 1.28 \end{pmatrix}$$

Calcul de $\mathbf{x}_3$:

$$\nabla f(\mathbf{x}_2) = \begin{pmatrix} 2 \cdot 1.28 \\ 2 \cdot 1.28 \end{pmatrix} = \begin{pmatrix} 2.56 \\ 2.56 \end{pmatrix}$$

$$\mathbf{x}_3 = \mathbf{x}_2 - \alpha \nabla f(\mathbf{x}_2) = \begin{pmatrix} 1.28 \\ 1.28 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 2.56 \\ 2.56 \end{pmatrix} = \begin{pmatrix} 1.024 \\ 1.024 \end{pmatrix}$$

Calcul de $\mathbf{x}_4$:

$$\nabla f(\mathbf{x}_3) = \begin{pmatrix} 2 \cdot 1.024 \\ 2 \cdot 1.024 \end{pmatrix} = \begin{pmatrix} 2.048 \\ 2.048 \end{pmatrix}$$

$$\mathbf{x}_4 = \mathbf{x}_3 - \alpha \nabla f(\mathbf{x}_3) = \begin{pmatrix} 1.024 \\ 1.024 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 2.048 \\ 2.048 \end{pmatrix} = \begin{pmatrix} 0.8192 \\ 0.8192 \end{pmatrix}$$

Calcul de $\mathbf{x}_5$:

$$\nabla f(\mathbf{x}_4) = \begin{pmatrix} 2 \cdot 0.8192 \\ 2 \cdot 0.8192 \end{pmatrix} = \begin{pmatrix} 1.6384 \\ 1.6384 \end{pmatrix}$$

$$\mathbf{x}_5 = \mathbf{x}_4 - \alpha \nabla f(\mathbf{x}_4) = \begin{pmatrix} 0.8192 \\ 0.8192 \end{pmatrix} - 0.1 \cdot \begin{pmatrix} 1.6384 \\ 1.6384 \end{pmatrix} = \begin{pmatrix} 0.65536 \\ 0.65536 \end{pmatrix}$$

Chaque itération réduit la valeur de la fonction f, nous rapprochant ainsi du minimum global qui est $(0, 0)$ pour cette fonction quadratique simple.

2.3.5.1 En Python

```python
import numpy as np
# Définition de la fonction et de ses dérivées partielles
def f(x, y):
    return x**2 + y**2

def grad_f(x, y):
    df_dx = 2 * x
    df_dy = 2 * y
    return np.array([df_dx, df_dy])

# Paramètres initiaux
x0 = np.array([2.0, 2.0]) # Point de départ (x, y)
alpha = 0.1  # Taux d'apprentissage
n_iterations = 50  # Nombre d'itérations

# Fonction de la descente de gradient
def gradient_descent(x0, alpha, n_iterations):
    x = x0
    history = [x0]  # Pour stocker l'évolution des points
    for i in range(n_iterations):
        gradient = grad_f(x[0], x[1])  # Calcul du gradient
        x = x - alpha * gradient  # Mise à jour des paramètres
        history.append(x)  # Stocker l'évolution des paramètres
        print(f"Iteration {i+1}: x = {x}, f(x, y) = {f(x[0], x[1])}")
    return x, history

# Exécution de l'algorithme de descente de gradient
final_x, history = gradient_descent(x0, alpha, n_iterations)

print("Point minimum approximé :", final_x)
print("Valeur de la fonction en ce point :", f(final_x[0], final_x[1]))
```

Le résultat de ce code est :
Point minimum approximé : [2.85449539e-05 2.85449539e-05]
Valeur de la fonction en ce point : 1.6296287810675901e-09

Ce code montre que la descente du gradient converge vers le minimum (qui est $(0, 0)$). Vous pouvez ajuster le taux d'apprentissage α ou le nombre d'itérations pour observer différents comportements.

Si on veut calculer aussi les dérivées partielles avec Python (**sympy**), on peut utiliser l'approche suivante :

```python
import sympy as sp
import numpy as np

# Définir les variables symboliques
x_sym, y_sym = sp.symbols('x y')

# Définir la fonction f(x, y)
f = x_sym**2 + y_sym**2

# Calcul des dérivées partielles (gradient)
df_dx = sp.diff(f, x_sym)
df_dy = sp.diff(f, y_sym)

```

```python
# Convertir les dérivées en fonctions lambdas pour évaluation numérique
grad_f_x = sp.lambdify([x_sym, y_sym], df_dx, 'numpy')
grad_f_y = sp.lambdify([x_sym, y_sym], df_dy, 'numpy')

# Fonction de descente du gradient
def gradient_descent(x0, alpha, n_iterations):
    x = np.array(x0, dtype='float64')  # Convertir x0 en tableau numpy pour permettre la mise à jour
    for i in range(n_iterations):
        # Calculer le gradient aux points (x[0], x[1])
        grad_x = grad_f_x(x[0], x[1])
        grad_y = grad_f_y(x[0], x[1])

        # Mise à jour des paramètres x et y
        x[0] = x[0] - alpha * grad_x
        x[1] = x[1] - alpha * grad_y

        # Afficher l'évolution
        x_val = x[0]
        y_val = x[1]
       # Utiliser les symboles SymPy dans `subs`
        print(f"Iteration {i+1}: x = {x_val}, y = {y_val}, f(x, y) = {f.evalf(subs={x_sym: x_val, y_sym: y_val})}")
    return x[0], x[1]

# Paramètres initiaux
x0 = [2.0, 2.0]  # Point de départ
alpha = 0.1  # Taux d'apprentissage
n_iterations = 50  # Nombre d'itérations

# Exécution de la descente du gradient
x_min, y_min = gradient_descent(x0, alpha, n_iterations)
print(f"Minimum approximé : x = {x_min}, y = {y_min}")
```

2.4 Fonctions et dérivées couramment utilisées en IA

Fonction	**Dérivée**
$f(x) = x^n$	$f'(x) = nx^{n-1}$
$f(x) = u^n(x)$	$f'(x) = n \times u'(x) \times (u(x))^{n-1}$
$f(x) = \ln(u(x))$	$f'(x) = \frac{u'(x)}{u(x)}$
$f(x) = e^{u(x)}$	$f'(x) = u'(x)\, e^{u(x)}$
$f(x) = \frac{1}{1+e^{-x}}$ (Sigmoïde)	$f'(x) = f(x)(1 - f(x))$
$f(x) = \arctan(u(x))$	$f'(x) = \frac{u'(x)}{1+(u(x))^2}$
$f(x) = \tanh(x)$	$f'(x) = 1 - \tanh^2(x)$
$f(x) = \max(0, x)$ (ReLU)	$f'(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{si } x \leq 0 \end{cases}$
$f(x) = \lvert x \rvert$	$f'(x) = \begin{cases} 1 & \text{si } x > 0 \\ -1 & \text{si } x < 0 \end{cases}$
$f(x) = \log(1 + e^x)$ (Softplus)	$f'(x) = \frac{1}{1+e^{-x}}$
$f(x) = \sum_i x_i \log x_i$ (Entropie)	$\frac{\partial f}{\partial x_i} = 1 + \log x_i$