L'Intelligence Artificielle repose fortement sur les mathématiques. Les algorithmes d'IA utilisent des modèles capables de représenter, analyser et transformer des données numériques. Parmi les outils mathématiques les plus importants figurent l'**algèbre linéaire**, les **probabilités** et les **statistiques**. Ces disciplines permettent de modéliser les données, d'optimiser les modèles et d'évaluer leurs performances.

**2.1 Algèbre linéaire**

L'algèbre linéaire constitue la base du Machine Learning et du Deep Learning. Elle permet de représenter les données et les paramètres des modèles sous forme de vecteurs et de matrices.

**2.1.1 Vecteurs**

Un **vecteur** est un ensemble ordonné de nombres. En IA, il représente souvent une observation ou un ensemble de caractéristiques (features).

Exemple :

$$x = (x_1, x_2, \dots, x_n)$$

Chaque composante correspond à une information (taille, poids, température, intensité, etc.).

Les opérations principales sur les vecteurs sont :

- Addition,
- Multiplication par un scalaire,
- Produit scalaire.

Le **produit scalaire** permet de mesurer la similarité entre deux vecteurs :

Le **produit scalaire** permet de mesurer la similarité entre deux vecteurs :

$$x \cdot y = \sum_{i=1}^{n} x_i y_i$$

**2.1.2 Matrices**

Une **matrice** est un tableau de nombres organisés en lignes et colonnes. En IA, elles servent à représenter des ensembles de données ou des transformations.

Exemple :

$$A \in \mathbb{R}^{mxn}$$

Les opérations principales sont :

- Addition,
- Multiplication,
- Transposition,
- Inversion (si possible).

La multiplication matricielle est essentielle pour calculer les sorties des réseaux neuronaux.

**2.1.3 Produits matriciels**

Le **produit matriciel** permet de combiner deux matrices compatibles :

$$C = A \times B$$

Il est utilisé dans :

- La propagation dans les réseaux neuronaux,
- La transformation des données,
- Les modèles linéaires.

Chaque neurone effectue essentiellement un produit entre un vecteur d'entrée et un vecteur de poids.

**2.1.4 Normes**

Une **norme** mesure la taille ou la longueur d'un vecteur.

Les plus courantes sont :

- Norme L1 :

$$\|x\|_1 = \sum |x_i|$$

- Norme L2 :

$$\|x\|_2 = \sqrt{\sum x_1^2}$$

- Norme infinie :

$$\|x\|_\infty = \max|x_i|$$

En IA, les normes sont utilisées pour :

- Mesurer l'erreur,
- Régulariser les modèles,
- Comparer des vecteurs.

**2.2 Probabilités et statistiques**

Les probabilités et statistiques permettent de gérer l'incertitude, d'analyser les données et d'évaluer les performances des modèles d'IA.

### 2.2.1 Variables aléatoires

Une **variable aléatoire** représente un phénomène dont la valeur dépend du hasard.

On distingue :

- Variables discrètes (nombre fini de valeurs),
- Variables continues (ensemble infini).

Exemple : nombre de défauts, température, prix.

### 2.2.2 Espérance mathématique

L'**espérance** est la valeur moyenne attendue d'une variable aléatoire.

Pour une variable discrète :

$$E(X) = \sum x_i P(x_i)$$

Pour une variable continue :

$$E(X) = \int x f(x) dx$$

En IA, elle sert à :

- Évaluer un modèle,
- Calculer une perte moyenne,
- Optimiser des algorithmes.

### 2.2.3 Variance et écart-type

La **variance** mesure la dispersion autour de la moyenne :

$$Var(X) = E[\big(X - E(X)\big)^2]$$

L'**écart-type** est la racine carrée de la variance.

$$\sigma = \sqrt{E[\big(X - E(X)\big)^2]}$$

Ces mesures permettent de comprendre la stabilité et la fiabilité des données ou des prédictions.

### 2.2.4 Lois de probabilité usuelles

#### a) Loi uniforme

Toutes les valeurs ont la même probabilité sur un intervalle.

Utilisée pour :

- L'initialisation des poids,

- La simulation.

**b) Loi binomiale**

Modélise le nombre de succès dans une suite d'essais.

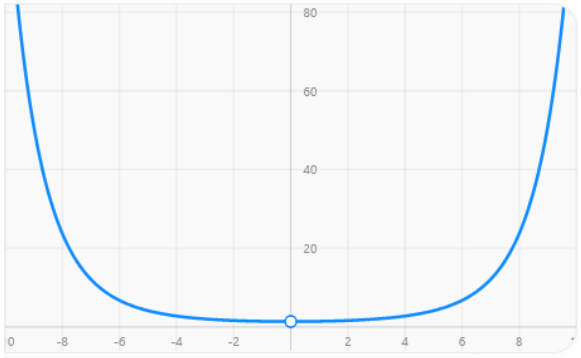
Utilisée pour :

- Classification,
- Estimation de performance.

**c) Loi normale (gaussienne)**

La plus utilisée en IA. Elle modélise de nombreux phénomènes naturels.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$




Utilisée pour :

- Bruit,
- Modélisation des données,
- Normalisation.

**2.3 Rôle des mathématiques dans l'IA**

Les mathématiques permettent :

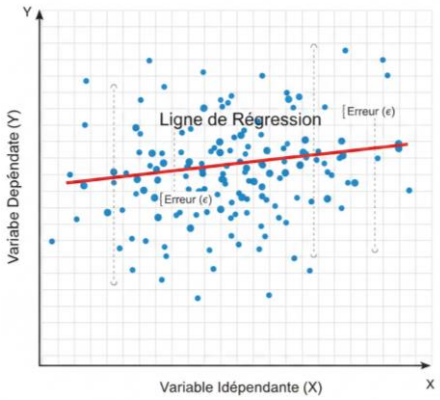
- De représenter les données,
- D'entraîner les modèles,
- D'optimiser les paramètres,
- D'évaluer les résultats,
- De garantir la robustesse.

Sans algèbre linéaire et statistiques, il n'existerait pas de Machine Learning ni de Deep Learning.

**3. Régression linéaire simple**

**3.1. Introduction et objectif**

La régression linéaire simple est l'un des modèles fondamentaux du Machine Learning supervisé. Elle permet d'établir une relation mathématique entre une **variable explicative** x et une **variable cible** y.




En mécanique et énergétique, elle sert par exemple à :

- Prédire une puissance à partir d'une vitesse,
- Estimer une consommation énergétique selon la charge,
- Modéliser une température en fonction du temps ou du débit,
- Relier une pression à un débit dans un système fluide.

Le but est d'apprendre automatiquement une loi du type :

$$y = f(x)$$

à partir de données expérimentales.

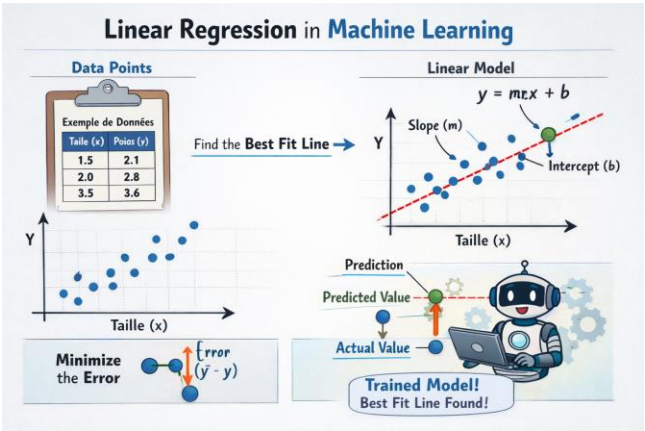
### 3.2. Formulation mathématique

La régression linéaire simple suppose que la relation entre x et y est **linéaire** :

$$y = wx + b$$

où :

- *w* est la pente (coefficient),
- *b* est le biais (ordonnée à l'origine),
- *x* est la variable d'entrée,
- *y* est la sortie prédite.



Pour un ensemble de données$\{(x_i, y_i)\}_{i=1}^{N}$ N, le modèle devient :

$$\hat{y}_i = wx_i + b$$

**3.3. Fonction de coût**

Pour mesurer l'erreur du modèle, on définit une **fonction de coût**. La plus utilisée est l'**erreur quadratique moyenne (MSE)** :

$$J(\omega, b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

avec :

$$\hat{y}_i = wx_i + b$$

Cette fonction pénalise fortement les grandes erreurs et permet une optimisation stable.

**Interprétation physique :**

- Si J est grand → le modèle est mauvais.
- Si J est petit → le modèle représente bien le phénomène réel.

On cherche donc :

$$\min_{w,b} J(\omega, b)$$

**3.4. Optimisation**

**a) Principe**

L'optimisation consiste à trouver les paramètres w et b qui minimisent la fonction de coût.

Deux méthodes principales existent :

- La solution analytique (moindres carrés),
- La descente de gradient.

---

**b) Descente de gradient**

La descente de gradient ajuste progressivement les paramètres :

$$w \coloneqq w - \alpha\frac{\partial J}{\partial w}$$

$$b \coloneqq b - \alpha\frac{\partial J}{\partial b}$$

Où :

- α est le taux d'apprentissage (learning rate),

- sont les gradients.

Les dérivées sont :

$$\frac{\partial J}{\partial w} = \frac{2}{N}\sum x_i(y_i - \hat{y}_i)$$

$$\frac{\partial J}{\partial b} = \frac{2}{N}\sum (y_i - \hat{y}_i)$$

Le processus est répété jusqu'à convergence.

**c) Sens physique**

À chaque itération :

- On mesure l'erreur,
- On corrige la pente,
- On corrige le biais,
- On améliore la prédiction.

Cela simule un ajustement progressif de la loi physique à partir des données.

**3.5. Mise en œuvre avec Scikit-learn**

Scikit-learn fournit une implémentation optimisée de la régression linéaire.

a) Importation

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

b) Exemple de données énergétiques

```
# Données : vitesse -> puissance
X = np.array([10, 20, 30, 40, 50]).reshape(-1,1)
y = np.array([15, 28, 45, 60, 80])
```

c) Séparation apprentissage / test

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

d) Création et entraînement

```
model = LinearRegression()
model.fit(X_train, y_train)
```

e) Paramètres du modèle

```
print("Pente w =", model.coef_[0])
print("Biais b =", model.intercept_)
```

f) Prédiction

```
y_pred = model.predict(X_test)
```

g) Visualisation

```
plt.scatter(X, y, label="Données réelles")
plt.plot(X, model.predict(X), color='red', label="Modèle")
plt.xlabel("Vitesse")
plt.ylabel("Puissance")
plt.legend()
plt.show()
```

### 3.6. Évaluation du modèle

On mesure la qualité du modèle avec MSE :

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
print("MSE =", mse)
```

### 3.7. Lien avec la mécanique et l'énergie

Applications typiques :

- Estimation de consommation électrique,
- Modélisation thermique,
- Rendement d'une machine,
- Lois débit-pression,
- Vitesse-puissance,
- Usure-temps.

La régression linéaire permet de transformer des mesures physiques en **modèles exploitables pour la prédiction et l'optimisation industrielle**.