Chapitre 2 : Mathématiques de base pour l'IA

1. **Algèbre linéaire** : vecteurs, matrices, produits, normes.
2. **Probabilités et statistiques** :
   - Variables, espérance, variance.
   - Lois usuelles : normale, binomiale, uniforme
3. **Régression linéaire simple :**
   - Formulation, coût, optimisation.
   - Mise en œuvre avec Scikit-learn
4. **Exercices :**
   - Manipulation de matrices avec la bibliotheque NumPy(Python)
   - Exercice sur la régression linéaire(utiliser une bibliothèque Python comme Scikit-learn par exemple)
   - Expliquer la bibliothèque Matplotlib(Python)

Ce chapitre constitue le socle indispensable pour comprendre comment les algorithmes d'IA (comme l'ANN ou le SVR que vous étudiez) traitent les données.

1. Algèbre linéaire

L'IA voit les données comme des objets géométriques.

- **Vecteurs :** Une liste de nombres $[x_1, x_2, \ldots, x_n]$ représentant une observation.
- **Matrices :** Un tableau de nombres (lignes x colonnes). C'est le format standard pour un jeu de données complet.
- **Produit matriciel :** Opération fondamentale du Deep Learning. Dans un neurone, la sortie est le produit des entrées par les poids : $Y = W.X + b$.
- **Normes :** Utilisées pour mesurer la "taille" d'un vecteur ou l'erreur du modèle. La **Norme L2** (Euclidienne) est la base du calcul du RMSE.

2. Probabilités et statistiques

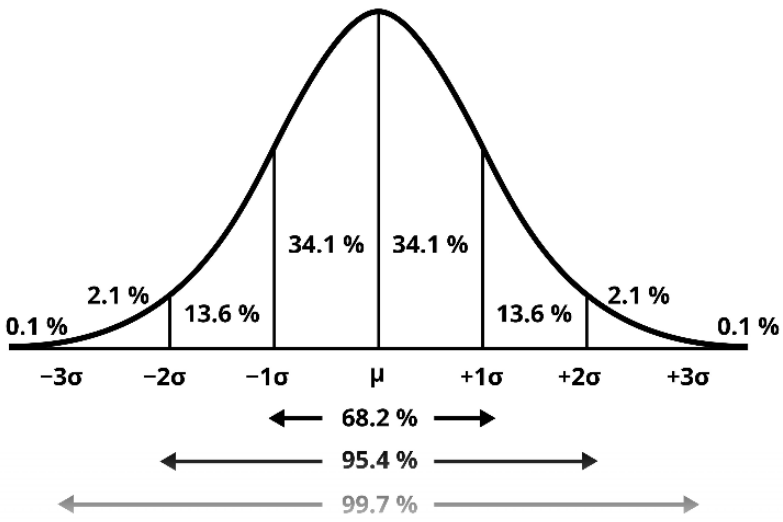

Figure : Densité de probabilité de la loi gaussienne

- **Variables, espérance, variance :**

- **Espérance (E[X]) :** La moyenne théorique.
- **Variance (σ^2):** Mesure la dispersion des données. Un modèle avec une variance élevée est en "Overfitting".

- **Lois usuelles :**
  - **Loi Normale (Gaussienne) :** La plus importante en IA. On suppose souvent que les erreurs de mesure suivent cette loi.
  - **Loi Binomiale :** Pour les problèmes de classification binaire (Oui/Non).
  - **Loi Uniforme :** Souvent utilisée pour initialiser les poids d'un réseau de neurones au hasard.

3. Régression linéaire simple

- **Formulation :** On cherche la droite d'équation $y = ax + b$.
- Fonction de coût (MSE) : On calcule l'écart entre le réel et la prédiction :

$$MSE = \frac{1}{n}\sum(y_{réel} - y_{pred})^2$$

- **Optimisation :** On utilise la **Descente de Gradient** pour ajuster a et b afin de minimiser le coût.

4. Exercices pratiques

Exercice 1 : Manipulation de matrices avec NumPy

```python
import numpy as np
# Création d'une matrice 2x3
A = np.array([[1, 2, 3], [4, 5, 6]])
# Produit matriciel
B = np.ones((3, 2))  # Matrice de 1
produit = np.dot(A, B)
print("Matrice A :\n", A)
print("Dimensions de A :", A.shape)
print("Résultat du produit :\n", produit)
```

Exercice 2 : Régression linéaire avec Scikit-learn

Python

```python
from sklearn.linear_model import LinearRegression
import numpy as np

# Données d'exemple
X = np.array([[1], [2], [3], [4]])
# Entrées
y = np.array([2, 4, 6, 8])
# Sorties (y = 2x)
# Créer et entraîner le modèle
model = LinearRegression()
model.fit(X, y)
# Prédire pour x = 5
prediction = model.predict([[5]])
print(f"Pour x=5, le modèle prédit y={prediction[0]}")
```

Exercice 3 : Expliquer la bibliothèque Matplotlib

Matplotlib est la bibliothèque de référence pour la visualisation de données en Python. Elle permet de créer des graphiques de haute qualité.

- **`plt.plot()` :** Pour dessiner des lignes (courbes de perte, tendances).
- **`plt.scatter()` :** Pour dessiner des nuages de points (données expérimentales).
- **`plt.xlabel()` / `plt.ylabel()` :** Pour nommer les axes (Indispensable !).

Résumé : Ce chapitre est le lien entre la théorie et la pratique. L'algèbre linéaire permet de structurer les données, les statistiques permettent d'évaluer la confiance du modèle, et la régression est la forme la plus simple d'apprentissage supervisé.