1. Introduction

A l'ère du Big Data, l'apprentissage automatique (machine Learning) est une discipline émergente pour développer une intelligence artificielle qui permet de :

- d'analyser automatiquement des données;
- de détecter des motifs et associations de données;
- d'utiliser des motifs pour la prédiction de données;
- de prendre des décisions automatiques;

L'apprentissage automatique est actuellement l'approche préférée pour les domaines de:

- Parole: reconnaissance de parole, personne.
- Vision artificielle: reconnaissance d'objets, segmentation, etc.
- Robotique: estimation de positions, de cartes, d'état, etc.
- Bio-informatique: alignement de séquences, analyse de données génétiques.
- E-commerce: commerce automatique, forage de données, spams.
- Analyse financière: allocation de portfolio, crédits, bourses, etc.
- Médicine: diagnostique, traitement, conception de thérapies.
- Graphisme: conception et simulations réalistes.
- Web: Gestion du contenu, entrepôts de données, réseaux sociaux, etc.

Il existe trois types d'apprentissage:

- **Apprentissage supervisé:** Il y a une cible à prédire.

$$D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$$

- **Apprentissage non-supervisé: la** cible n'est pas fournie.

$$D = \{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$$

- **Apprentissage par renforcement:** ensemble des méthodes qui permettent à un agent d'apprendre à choisir quelle action prendre, et ceci de manière autonome.

Apprentissage supervisé: Classification, régression.

- **Classification:** La cible est un indice de classe $y \in \{1, \dots, K\}$.

  Exemple: reconnaissance de caractères manuscrits.

  x: valeurs des intensités des pixels de l'image. y: identité du caractère (classe).

- **Régression:** La cible est un nombre réel $y \in \mathbb{R}$.

Exemple: Prédiction de la valeur d'une action de bourse. x: Informations économiques de la journée. y: Valeur de l'Actions en bourse (nombre réeel).

Dans ce cas, on a :

- Un exemple d'apprentissage a la forme $(x^{(i)}, y^{(i)})$, où:

$$x^{(i)} = (x_1{}^{(i)}, x_2{}^{(i)}, \ \ldots, x_D{}^{(i)})$$

- D est le nombre d'attributs.
- L'ensemble d'entrée *(les observations)* contient N exemples.
- On dénote par X l'espace des variables d'entrées (ex. $\mathbb{R}^D$)
- On dénote par Y l'espace des variables de sortie (ex. $\mathbb{R}$)

2. Régression Linéaire

2.1. Exemple introductif

Soit les différentes valeurs prises par une variable X auxquelles correspondent les valeurs de y. Les valeurs de X que l'on note : $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, $x^{(4)}$, $x^{(5)}$, ou plus généralement $x^{(i)}$, pour une $i^{ème}$ instance peuvent représenter la surface d'un appartement (en $10\times m^2$), les $y^{(i)}$ représentent le prix de l'appartement correspondant (en 10^6 DA). On souhaite alors prédire la valeur d'un nouvel appartement dont on connaît la surface.

X=[1, 2, 3, 4, 5]

Y=[2, 3, 5, 4, 6]

Pour celà, on va construire un modèle, qui est une fonction : $f : X \rightarrow y$ qui permet de prédire la valeur de y sachant la surface de l'appartement. Le modèle que l'on se propose de construire est un modèle linaire, on parle alors : **de Régression Linéaire**. C'est-à-dire :

$$y = f(x) = \ aX + b$$

On notera ici, que dans le cas général, un exemple $x^{(i)}$ peut être décrit par une ou plusieurs caractéristiques (features), dans notre exemple, on peut considérer la surface de l'appartement et celle des parties communes, le modèle s'écrit donc de manière générale.

$$y = f(x) = \ w_0 + x_1 w_1 + x_2 w_2 + \cdots$$

Cette écriture se traduit en notation matricielle, comme étant :

$$y = f(x) = wX$$

Où, y est le vecteur qui regroupe les différentes valeurs $y^{(i)}$, w est un vecteur qui regroupe les poids w_j, et X une matrice qui comprend les différentes valeurs des $x^{(i)}$ auxquelles, on rajoute une colonne de 1 qui prend en charge la valeur du biais w_0 au travers d'une valeur x_0=1 pour toutes les $x^{(i)}$.

Dans notre exemple, on obtient :

$$X = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix} \qquad y = \begin{bmatrix} 2 \\ 3 \\ 5 \\ 4 \\ 6 \end{bmatrix}$$

2.2.Méthode matricielle

Pour trouver le modèle, il faut calculer w_0 et w_1 qui représentent respectivement, la pente et le biais de notre droite de régression linéaire. La solution analytique proposée est une méthode matricielle appelée (OLS pour Ordinary Least Squares) car elle est basée sur la minimisation de l'erreur quadratique, c'est-à-dire la différence entre les valeurs observées et les valeurs prédites de y. Cette erreur est appelée la moyenne de l'erreur quadratique (MSE pour Mean Square Error).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(f\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Pour notre cas,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\left(w_1 x^{(i)} + w_0\right) - y^{(i)}\right)^2$$

Cette solution matricielle permet de trouver *w* selon la formule :

$$\widehat{w} = (X^T X)^{-1} X^T y$$

On commence nos calculs :

a. Calculer X^TX :

$$X^T X = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix} = \begin{bmatrix} 55 & 15 \\ 15 & 5 \end{bmatrix}$$

b. Calcul de X^Ty

$$X^T y = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 5 \\ 4 \\ 6 \end{bmatrix} = \begin{bmatrix} 69 \\ 20 \end{bmatrix}$$

c. Inverse X^TX

Pour une matrice 2×2, on a :

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

D'où :

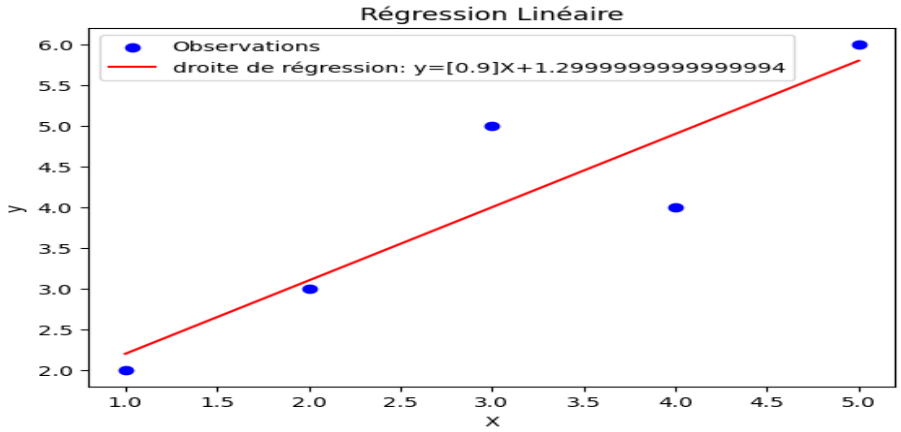
$$(X^T X)^{-1} = \begin{bmatrix} 55 & 15 \\ 15 & 5 \end{bmatrix}^{-1} = \frac{1}{50}\begin{bmatrix} 5 & -15 \\ -15 & 55 \end{bmatrix} = \begin{bmatrix} 0.1 & -0.3 \\ -0.3 & 1.1 \end{bmatrix}$$

d. Calcul des poids

$$\widehat{w} = (X^T X)^{-1} X^T y = \begin{bmatrix} 0.1 & -0.3 \\ -0.3 & 1.1 \end{bmatrix} \begin{bmatrix} 69 \\ 20 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 1.3 \end{bmatrix}$$

Notre fonction *f(x)* prend donc l'expression : $f(x) = 0.9x + 1.3$

C'est d'ailleurs ce résultat que l'on obtient si on se réfère à la classe ***LinearRegression*** du module ***linear_model*** depuis la librairie ***sklearn*** de python. Comme illustré par la figure suivante.



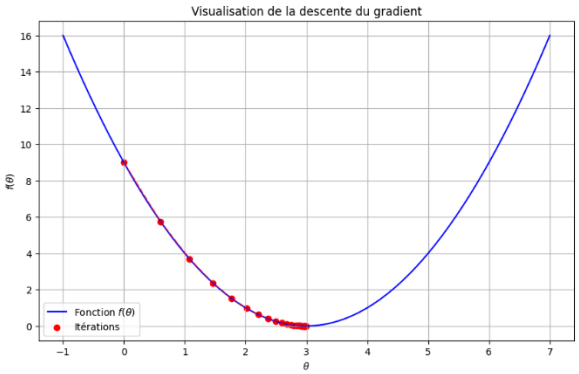
L'application de cette méthode n'est pas toujours possible, car si le nombre de données augmente la manipulation des matrices induit un temps de calcul très important. Plus problématique est le calcul de la matrice inverse qui n'est pas toujours possible car il requiert qu'un certains nombre de critères soient satisfaits.

2.3. La descente du gradient

On remarque que la fonction à minimiser est une fonction d'ordre 2 (carrés des erreurs).

$$J(a,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2$$

Pour minimiser la fonction de coût, il suffit de trouver le point où le gradient s'annule. Pour cela, l'algorithme de la descente du gradient (Gradient Descent) calcule le gradient de l'erreur. Et en partant de valeur aléatoire de a et de b, l'algorithme va les mettre à jour progressivement pour s'approcher du point le plus bas, comme illustré dans la figure ci-dessous.



La descente du gradient est une méthode d'optimisation qui permet de trouver les meilleurs paramètres *a* et *b* en réduisant progressivement la fonction de coût.

Étapes principales :

1. Initialisation : on commence avec des valeurs aléatoires pour *a* et *b*.
2. Calcul du gradient : on calcule les dérivées partielles de la fonction de coût par rapport à *a* et *b*.
3. Mise à jour des paramètres :

$$a = a - \alpha.\frac{\partial J}{\partial a} \qquad b = b - \alpha.\frac{\partial J}{\partial b}$$

Avec :

$$\frac{\partial J}{\partial a} = -\frac{2}{n}\sum_{i=1}^{n} x_i(y_i - (ax_i + b))$$

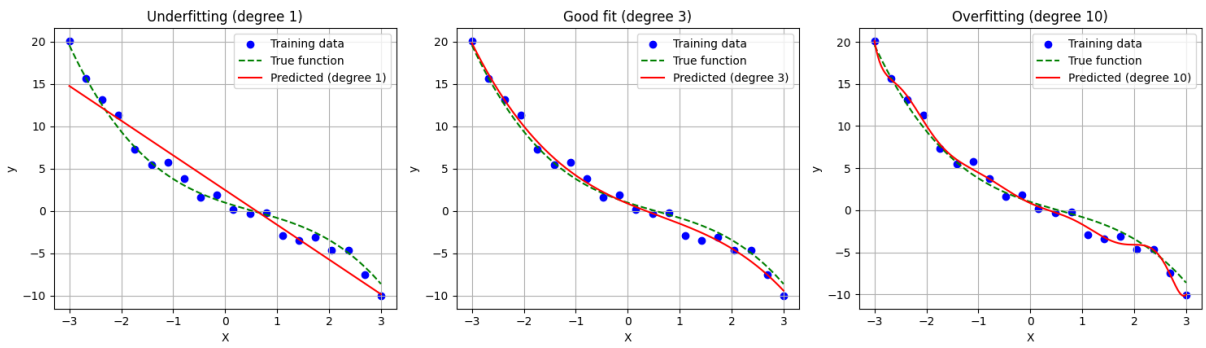
$$\frac{\partial J}{\partial b} = -\frac{2}{n}\sum_{i=1}^{n} (y_i - (ax_i + b))$$

Où α est le taux d'apprentissage (learning rate), un petit nombre (ex : 0.01).

Itération : on répète les étapes 2 et 3 jusqu'à ce que le coût devienne très petit (ou jusqu'à un nombre de cycles défini).

2.4.Overfitting vs underfitting

La figure suivante illustre deux des problèmes rencontrés lors de l'utilisation des méthodes d'apprentissage automatique.


Underfitting vs Good Fit vs Overfitting in Polynomial Regression
Underfitting (degree 1)
Good fit (degree 3)
Overfitting (degree 10)
Training data
True function
Predicted (degree 1)
Predicted (degree 3)
Predicted (degree 10)
X
y