Projet analyse de données

TP2 : Régression linéaire simple

Sarra Tajouri

Qu'est-ce que la régression linéaire simple ?

- Une méthode d'apprentissage supervisé.
- Données : une variable explicative et une variable à prédire

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \in \mathbb{R}^{m \times 1}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \in \mathbb{R}^m$$

Objectif :

- Trouver le coefficient β tel que $y = X\beta$.

Applications :

- Prévisions (ex. : prix, revenus).
- Analyse de tendance.
- Modélisation de relations simples (en économie, sociologie, biologie ...)

Objectifs du TP

Dans ce TP, nous allons explorer trois approches pour résoudre une régression linéaire simple :

- Méthode directe : solution au problème d'optimisation de moindres carrés.
- Utilisation de la fonction `minimize` de la bibliothèque `scipy.optimize`.
- Algorithme de descente de gradient.

Méthode analytique

- Trouver les coefficients β de la relation $y = X\beta$.
- Estimateur des moindres carrés :

$$\min_{\beta \in \mathbb{R}} \|X\beta - y\|_2^2 = \sum_{i=1}^{m} (x_i\beta - y_i)^2.$$

La solution est alors donnée par :

$$\boxed{\beta = (X^\top X)^{-1} X^\top y,}$$

à condition que $X^\top X$ soit inversible.

Avantages :

- Solution rapide et directe.
- Pas besoin d'initialisation.

Méthode analytique pour cas simple

Pour un modèle $y = w_1 x + w_0$, l'objectif est de minimiser la somme des erreurs quadratiques :

$$\frac{1}{m} \sum_{i=1}^{m} ((w_1 x_i + w_0) - y_i)^2$$

Les solutions analytiques pour w_1 et w_0 sont :

$$w_1 = \frac{\sum_{i=1}^{m} x_i y_i}{\sum_{i=1}^{m} x_i^2} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

Utilisation de `scipy.optimize.minimize`

Description :

- Formuler la régression comme un problème d'optimisation.
- Fonction coût : l'erreur quadratique moyenne (Mean Squared Error MSE).
- Utilisation de la fonction `minimize` pour optimiser les paramètres β.

Étapes :

1. Définir la fonction coût :

$$\text{MSE}(\beta) = \frac{1}{n}\|y - X\beta\|^2.$$

2. Initialiser les paramètres.
3. Appeler `minimize` pour trouver les valeurs optimales.

Idée de la descente de gradient

- Une méthode d'optimisation **itérative** pour **minimiser** une fonction coût.
- Utilisée dans de nombreux algorithmes d'apprentissage automatique.
- Pour des problèmes où une solution analytique n'est pas disponible.

Idée principale :

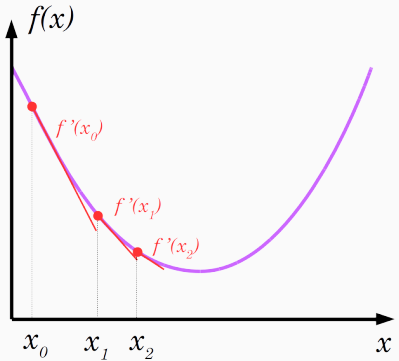
- Partir d'une estimation initiale des paramètres.
- Ajuster les paramètres en suivant la **pente** (gradient) négative de la fonction coût.
- Répéter jusqu'à convergence vers le minimum de la fonction coût.

Algorithme de descente de gradient

- Approche itérative pour minimiser la fonction coût.
- Mise à jour des paramètres à chaque itération :

$$x_{i+1} \leftarrow x_i - \alpha \nabla_\beta f_{\text{coût}},$$

où α est le pas d'apprentissage.



Algorithme : Descente de gradient

Input : α, $n_{\text{iter\_max}}$, ϵ
Output : Coefficients optimaux β

Initialiser β
for $k \leftarrow 1$ **to** n_{iter_max} **do**
    Calculer le gradient de la fonction coût : $\nabla_\beta f_{\text{coût}}$
    Mettre à jour les paramètres : $\beta \leftarrow \beta - \alpha \cdot \nabla_\beta f_{\text{coût}}$
    if $||\nabla f_{coût}|| < \epsilon$ *(e.g. Convergence vérifiée,)* **then**
        Break
    end
end
return β

Avantages de la descente de gradient

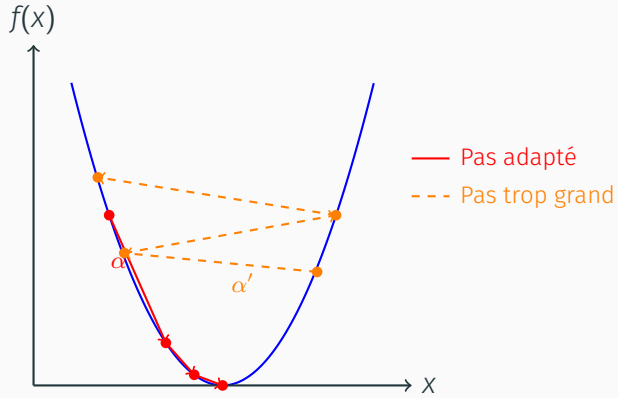
Comparaison avec le calcul de l'inverse d'une matrice :

- **Évolutivité :** Plus adaptée aux grandes dimensions (n très grand).
- **Mémoire efficace :** Peut être réalisée par mini-batch ou incrémentalement, limitant l'utilisation de la mémoire.
- **Adaptée aux problèmes non linéaires :** Convient pour des fonctions coût complexes où aucune solution analytique n'existe.
- **Contrôle de la précision :** Possibilité de s'arrêter à une solution suffisamment précise sans surcoût.

Facteurs influant la descente de gradient

1. Pas d'apprentissage α :

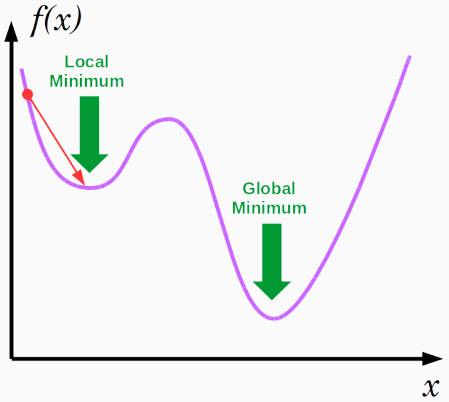
   - Contrôle la taille des pas dans la direction du gradient.
   - Choix critique pour assurer la convergence.



Facteurs influant la descente de gradient

2. Initialisation des paramètres :

- Une bonne initialisation peut accélérer la convergence.
- Mauvaise initialisation peut faire rester bloquer sur un minimum local.



Facteurs influant la descente de gradient

3. Nombre d'itérations :

- Déterminé par $n_{\text{iter\_max}}$
- Trop peu d'itérations peuvent donner une solution sous-optimale.

Conclusion

- Nous avons exploré trois façons de résoudre une régression linéaire simple.
- Chaque méthode a ses avantages et inconvénients selon le contexte.
- **Consignes** :
  - Faire le TP *Régression Linéaire et Optimisation* et le déposer sur nextcloud à la fin
  - Remplir le notebook sur la régression linéaire simple en utilisant les librairies python