

IA & Machine Learning (M354)



Ch4. Apprentissage Supervisé - Régression linéaire



Rappel sur l'apprentissage supervisé



Les 4 notions fondamentales de l'apprentissage supervisé



La régression Linéaire

Rappelle

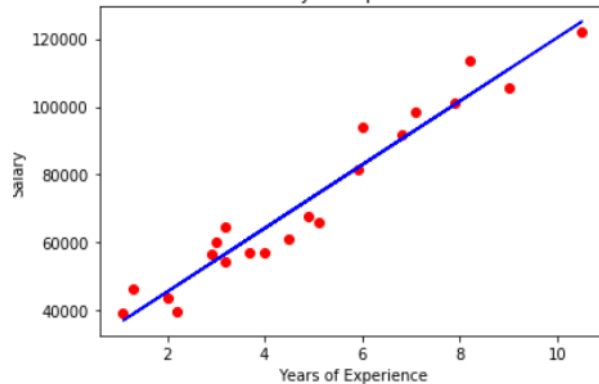
Apprentissage supervisé :

Une approche fondamentale du Machine Learning permettant à une machine d'apprendre à partir d'exemples étiquetés pour faire des prédictions.

Les taches d'apprentissage supervisé

Regression

Salary vs Experience



Regression Data

X ₁	X ₂	X ₃	X _p	Y
				5.2
				1.3
				23.0
				7.4

Numeric
Target

Nbr d'années
d'expérience

AI

Salaire

Les 4 notions fondamentales



Dataset

Ensemble d'exemples étiquetés composé de variables cibles (y) et de caractéristiques (X)



Modèle

Fonction mathématique avec des paramètres qui associe les caractéristiques aux prédictions



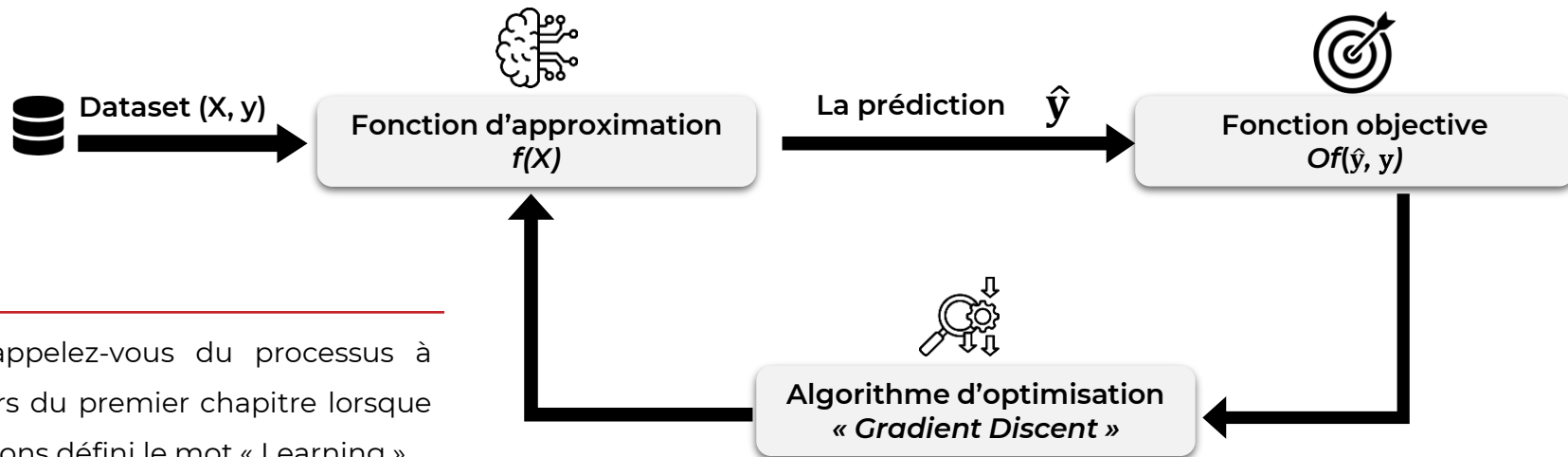
Fonction Coût

Mesure des erreurs entre les prédictions du modèle et les valeurs réelles



Algorithme de Minimisation

Stratégie pour trouver les paramètres optimaux qui minimisent la fonction coût



vous rappelez-vous du processus à côté, lors du premier chapitre lorsque nous avons défini le mot « Learning »

Les 4 notions fondamentales



Dataset

Ensemble d'exemples étiquetés utilisés pour entraîner un modèle d'apprentissage supervisé.



Variables cibles (y) : Les valeurs que nous voulons prédire, également appelées « étiquettes » ou « labels ».



Caractéristiques (X) : Les facteurs ou attributs (features) qui influencent la variable cible.
Ce sont les entrées utilisées pour faire des prédictions



Relation fondamentale : L'apprentissage supervisé cherche à établir une relation $y = f(X)$ à partir des exemples du dataset

y	x_1	x_2	x_3	...	x_n
$y^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$...	$x_n^{(1)}$
$y^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$...	$x_n^{(2)}$
$y^{(3)}$	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$...	$x_n^{(3)}$
...
$y^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$...	$x_n^{(m)}$
target	features				

Les 4 notions fondamentales

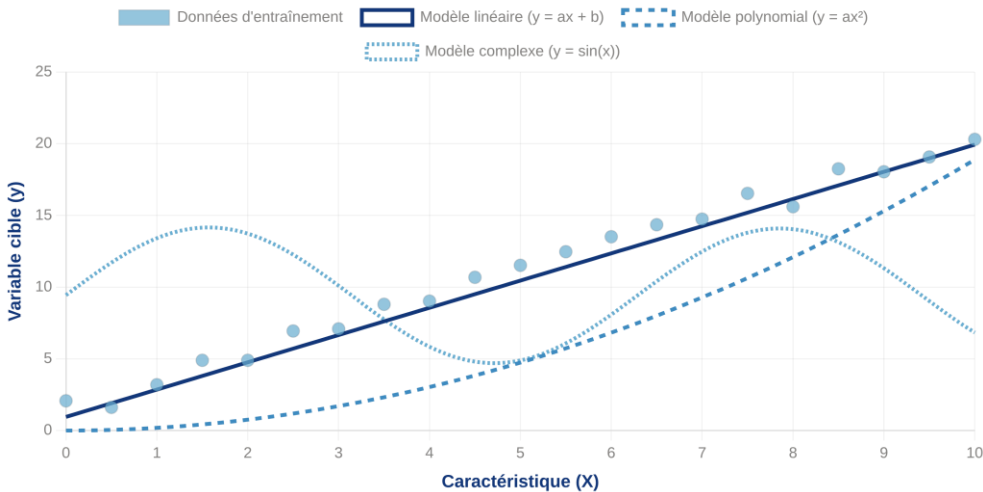


Modèle

Le modèle est une fonction mathématique qui associe les caractéristiques (X) aux prédictions de la variable cible (y).

↳ **Paramètres** : Les paramètres du modèle sont les valeurs que la machine doit apprendre à partir des données pour optimiser les prédictions

↳ **Sélection du modèle** : Notre travail consiste à sélectionner le modèle approprié, tandis que la machine s'occupe de trouver les paramètres optimaux



Modèle mathématique

$$y = \beta_0 + \beta_1 x + \varepsilon$$

y : variable dépendante (à prédire)

x : variable indépendante (explicative)

β_0 : ordonnée à l'origine (intercept)

β_1 : pente (coefficient directeur)

ε : terme d'erreur (résidu)

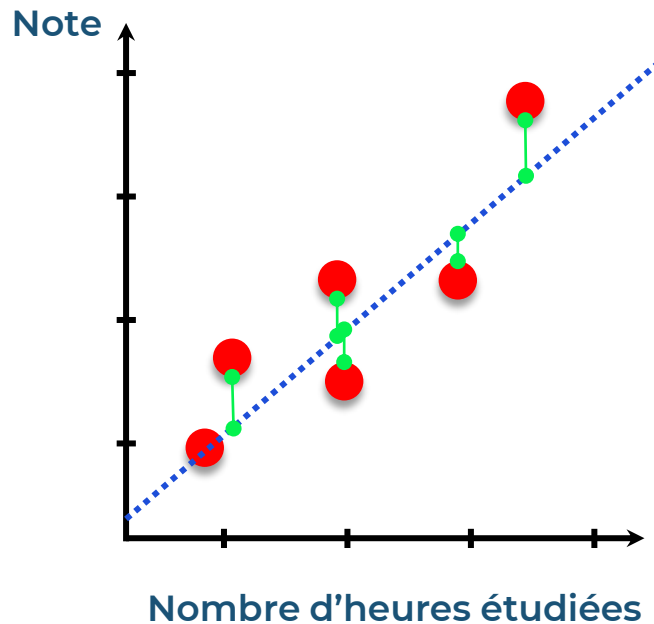
Les 4 notions fondamentales



Fonction coût

La fonction coût mesure l'écart entre les prédictions du modèle et les valeurs réelles du dataset, permettant d'évaluer la performance du modèle

- ↳ **Erreur Quadratique Moyenne** : Pour la régression linéaire, on utilise généralement l'erreur quadratique moyenne (MSE) qui élève au carré les différences pour pénaliser les grands écarts
- ↳ **Objectif d'Optimisation** : Un bon modèle minimise la fonction coût, ce qui signifie que ses prédictions sont proches des valeurs réelles du dataset



$$MSE = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i - y_i)^2$$

L'objectif est de trouver les valeurs des paramètres qui **minimisent** cette fonction de coût.

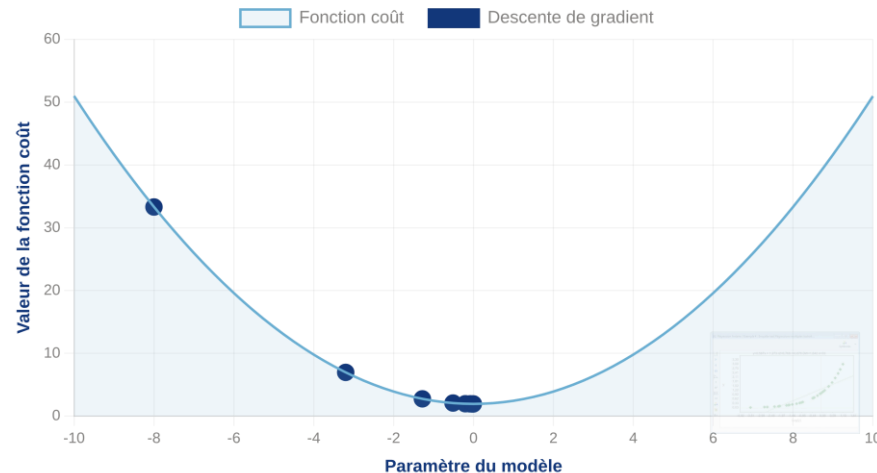
Les 4 notions fondamentales



Algorithme de Minimisation

Trouver les paramètres du modèle qui minimisent la fonction coût, c'est-à-dire qui réduisent l'erreur entre les prédictions et les valeurs réelles.

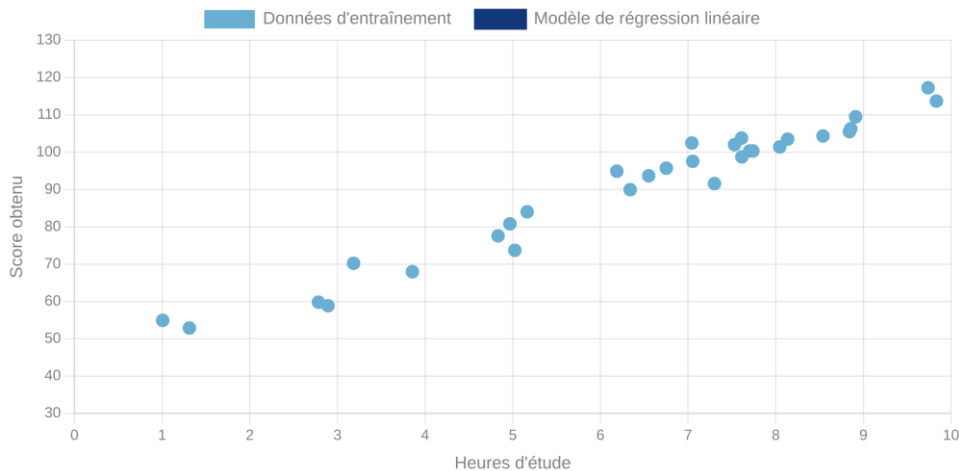
- ↳ **Méthode des moindres carrés** : Approche analytique qui cherche le point où la dérivée de la fonction coût est nulle, indiquant un minimum. Idéale pour la régression linéaire simple
- ↳ **Descente de gradient** : Algorithme itératif qui part d'un point aléatoire, calcule la pente (dérivée) de la fonction coût, puis se déplace par petits pas dans la direction opposée à la pente pour converger vers le minimum



Régression Linéaire

Un modèle fondamental d'apprentissage supervisé

La régression linéaire est le modèle le plus simple, mais elle constitue la base de nombreux modèles plus complexes en apprentissage supervisé.



Exemple:

Relation entre le nombre d'heures d'étude et les scores obtenus

Relation Linéaire

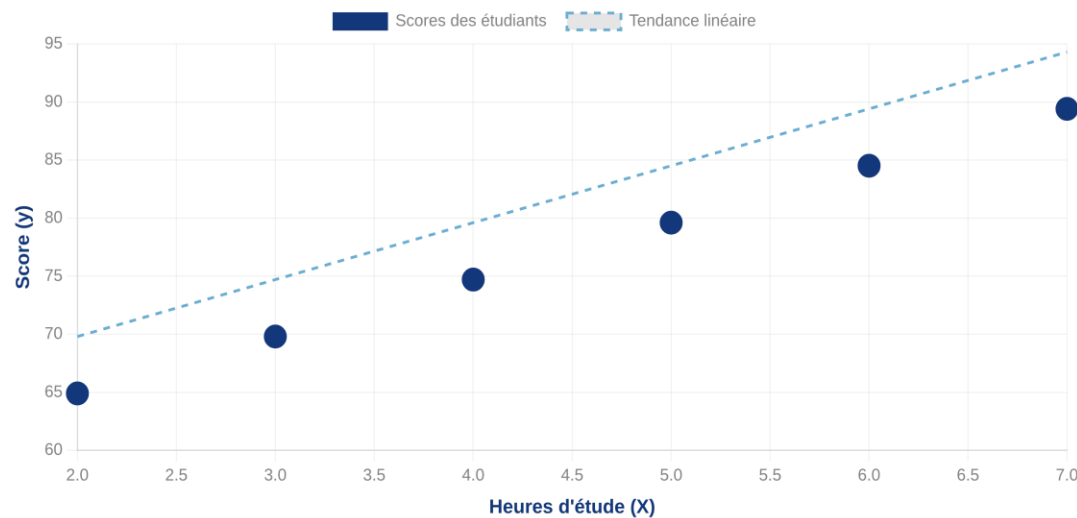
Elle modélise la relation entre une variable dépendante (y) et une ou plusieurs variables indépendantes (X) sous forme d'équation linéaire: $y = ax + b$.

Application des 4 Concepts

La régression linéaire illustre parfaitement les 4 notions fondamentales de l'apprentissage supervisé: dataset, modèle, fonction coût et algorithme d'optimisation.

Régression Linéaire : Dataset

Un modèle fondamental d'apprentissage supervisé



Exemple d'un dataset pour la régression linéaire:

Heures d'étude vs Score

Structure du Dataset

Le dataset contient 6 étudiants avec leurs heures d'étude (X) et leurs scores obtenus (y).

Variable Cible (y)

Le score de l'étudiant est notre variable cible que nous cherchons à prédire.

Caractéristique (X)

Le nombre d'heures d'étude est notre unique caractéristique (feature) dans cet exemple simple.

Relation Linéaire

On observe une tendance linéaire entre les heures d'étude et les scores, ce qui justifie l'utilisation d'un modèle de régression linéaire.

Régression Linéaire : Modèle

Un modèle fondamental d'apprentissage supervisé

Équation du modèle $f(x) = ax + b$

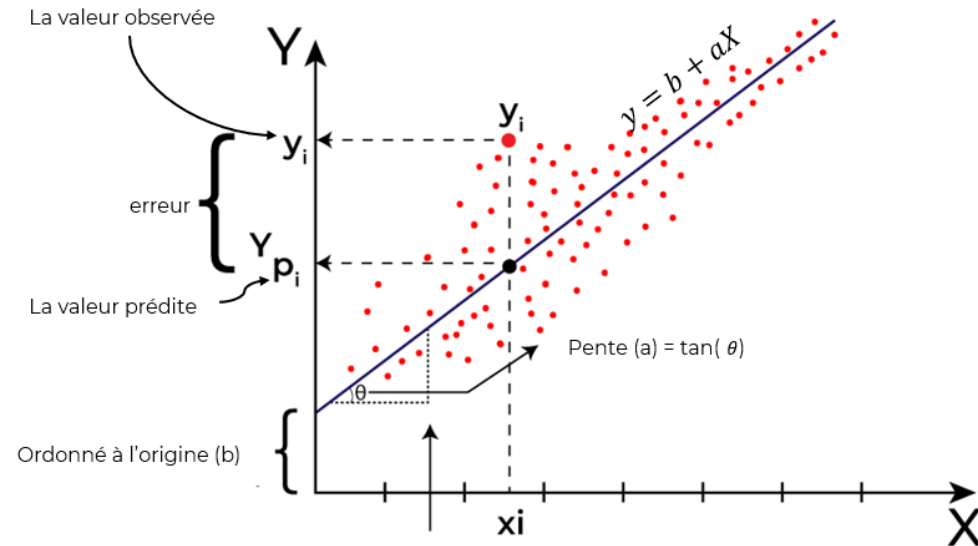
Le modèle de régression linéaire simple est défini par une équation linéaire où a est le coefficient directeur (pente) et b est l'ordonnée à l'origine.

Paramètres du Modèle a et b

Ces paramètres sont initialement inconnus et doivent être appris par la machine à partir des données d'entraînement pour minimiser les erreurs de prédiction.

Initialisation – Valeurs aléatoires

Le processus d'apprentissage commence avec des valeurs aléatoires pour les paramètres a et b , puis les ajuste progressivement pour améliorer les prédictions.



Régression Linéaire : Fonction Coût

Un modèle fondamental d'apprentissage supervisé

Erreur pour paramètre a trop petit



Erreur pour paramètre b trop petit



Erreur pour paramètre a trop grand



Erreur pour paramètres optimaux



Définition de la fonction coût

Pour la régression linéaire, on utilise l'erreur quadratique moyenne (MSE)

$$MSE = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i - y_i)^2$$

Pourquoi élever au carré?

L'élévation au carré des erreurs permet de pénaliser davantage les grandes erreurs et d'obtenir une fonction coût différentiable nécessaire pour l'optimisation.

Forme parabolique

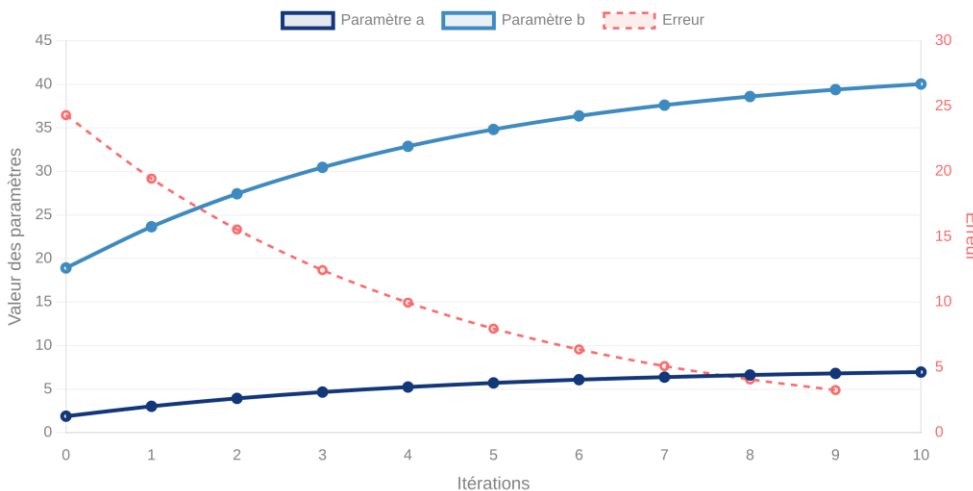
La fonction coût de la régression linéaire a une forme parabolique avec un minimum global unique, ce qui facilite l'optimisation des paramètres a et b.

Objectif d'optimisation

L'objectif est de trouver les valeurs de a et b qui minimisent cette fonction coût, c'est-à-dire qui réduisent au maximum l'écart entre les prédictions et les valeurs réelles.

Régression Linéaire : Optimisation

Un modèle fondamental d'apprentissage supervisé



Méthode des Moindres Carrés

Solution analytique qui trouve directement les valeurs optimales de a et b en résolvant les équations où les dérivées partielles de la fonction coût sont nulles.

Descente de Gradient

Méthode itérative qui ajuste progressivement les paramètres a et b en suivant la direction opposée au gradient de la fonction coût jusqu'à convergence.

Convergence

Avec un taux d'apprentissage approprié, la descente de gradient converge vers les mêmes valeurs optimales que la méthode des moindres carrés, mais de manière itérative.

moindres carrés ordinaires

Nombre d'heures étudiées	2	3.9	4	5.8	7
Score (Note)	9	12	7.5	12.5	18

X	Y	XY	X ²
2	9	18	4
3.9	12	46.8	15.21
4	7.5	29.6	16
5.8	12.5	72.5	33.64
7	18	126	49

$$\sum X = 22.7$$

$$\sum Y = 59$$

$$\sum XY = 292.9$$

$$\sum X^2 = 117.85$$

Score Prédit = ordonnée à l'origine + pente × Nombre d'heures étudiées

$$y = b + ax$$

$$a = \frac{n * \sum xy - \sum x * \sum y}{n * \sum x^2 - (\sum x)^2}$$

$$b = \frac{\sum y - a * \sum x}{n}$$

moindres carrés ordinaires : Démonstration

1. La fonction à minimiser (fonction de coût)

$$SSR = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (b + ax_i))^2$$

2. Conditions d'optimalité – dérivées partielles

$$\frac{\partial SSR}{\partial b} = -2 \sum_{i=1}^n (y_i - b - ax_i) = 0$$

$$\frac{\partial SSR}{\partial a} = -2 \sum_{i=1}^n x_i (y_i - b - ax_i) = 0$$

$$\sum_{i=1}^n y_i = nb + a \sum_{i=1}^n x_i$$

1

$$\sum_{i=1}^n x_i y_i = b \sum_{i=1}^n x_i + a \sum_{i=1}^n x_i^2$$

2

3. Résoudre les équations (1) et (2)

À partir de l'équations (1)

$$b = \frac{\sum_{i=1}^n y_i - a \sum_{i=1}^n x_i}{n}$$

3

Substituons (3) dans (2)

$$\sum_{i=1}^n x_i y_i - b \sum_{i=1}^n x_i = a \sum_{i=1}^n x_i^2 - b \sum_{i=1}^n x_i$$

moindres carrés ordinaires : Démonstration

3. Résoudre les équations (1) et (2)

À partir de l'équations (1)

$$b = \frac{\sum_{i=1}^n y_i - a \sum_{i=1}^n x_i}{n}$$

Substituons (3) dans (2)

$$\begin{aligned} \sum_{i=1}^n x_i y_i - a \sum_{i=1}^n x_i^2 &= b \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i y_i - a \sum_{i=1}^n x_i^2 &= \frac{\sum_{i=1}^n y_i - a \sum_{i=1}^n x_i}{n} \times \sum_{i=1}^n x_i \\ n \times \left(\sum_{i=1}^n x_i y_i - a \sum_{i=1}^n x_i^2 \right) &= \sum_{i=1}^n x_i \times \sum_{i=1}^n y_i - a \left(\sum_{i=1}^n x_i \right)^2 \\ n \times \sum_{i=1}^n x_i y_i - a n \times \sum_{i=1}^n x_i^2 &= \sum_{i=1}^n x_i \times \sum_{i=1}^n y_i - a \left(\sum_{i=1}^n x_i \right)^2 \\ n \times \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \times \sum_{i=1}^n y_i &= a \left(n \times \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2 \right) \\ a &= \frac{n \times \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \times \sum_{i=1}^n y_i}{n \times \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2} \end{aligned}$$

$$b = \frac{\sum y - a * \sum x}{n}$$

$$a = \frac{n * \sum xy - \sum x * \sum y}{n * \sum x^2 - (\sum x)^2}$$

moindres carrés ordinaires

Score Prédit = ordonnée à l'origine + pente × **Nombre d'heures étudiées**

$$y = b + ax$$

$$a = \frac{n * \sum xy - \sum x * \sum y}{n * \sum x^2 - (\sum x)^2} = \frac{5 * 292.9 - 22.7 * 59}{5 * 117.85 - 22.7^2} = 1.692807$$

$$b = \frac{\sum y - a * \sum x}{n} = \frac{59 - (1.692807) * 22.7}{5} = 4.11465622$$

$$y = 4.11465622 + 1.692807 \times x$$

$$\sum X = 22.7$$

$$\sum Y = 59$$

$$\sum XY = 292.9$$

$$\sum X^2 = 117.85$$

Nombre d'heures étudiées	2	3.9	4	5.8	7
Score (Note)	9	12	7.5	12.5	18
Score (Note) pred	7.5	10.7	10.8	13.9	15.96

moindres carrés ordinaires

Exercice

➤ En utilisant le dataset suivante, trouvez la droite de la régression qui ajuste les données pour prédire le prix de la maison

	Size	Price
0	2104	399900
1	1600	329900
2	2400	369000
3	1416	232000
4	3000	539900

$$m = \frac{n * \sum xy - \sum x * \sum y}{n * \sum x^2 - (\sum x)^2}$$

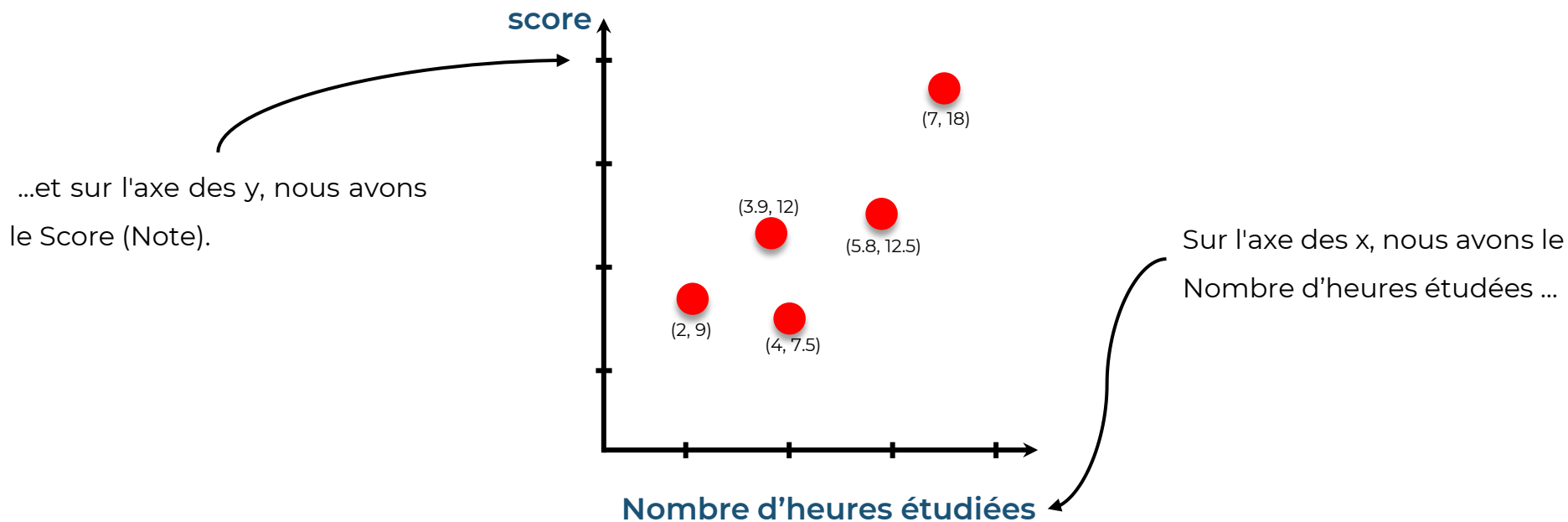
$$b = \frac{\sum y - m * \sum x}{n}$$

$$y = b + mx$$

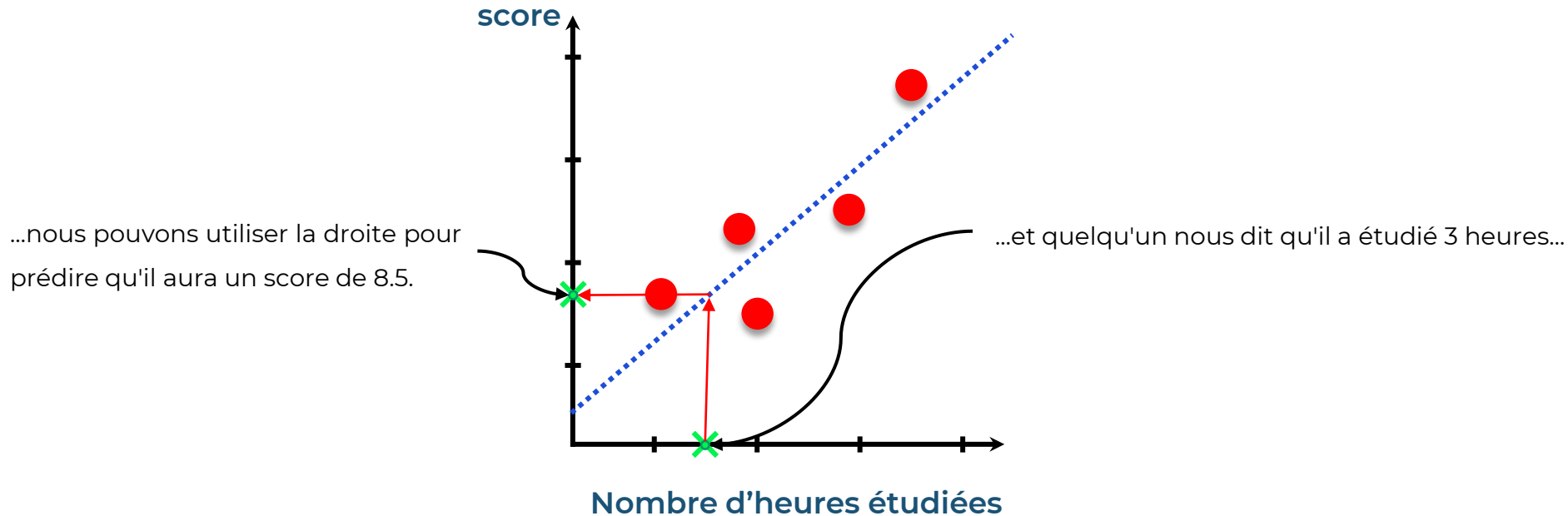
PS: jai pas bien compris cette partie qui concerne “data transformation” mes reponses sont de chatgpt donc je vais pas terminer

Descente de gradient

Commençons donc par l'exemple que nous avons déjà vu dans le 1^{er} chapitre un simple ensemble de données.



Descente de gradient



Score Prédit = ordonnée à l'origine + pente × **Nombre d'heures étudiées**

$$\text{Score Prédit} = b + a \times \text{Nombre d'heures étudiées}$$

Apprenons donc comment la **Descente de Gradient** peut ajuster une droite aux données en trouvant les valeurs optimales pour l'**Ordonnée à l'origine (b)** et la **Pente (a)**.

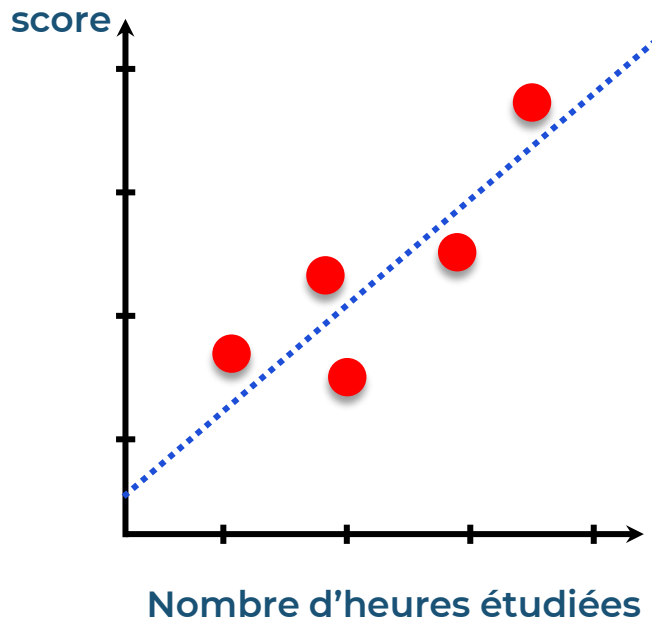
Descente de gradient

$$\text{Score Prédit} = \text{ordonnée à l'origine} + \text{pente} \times \text{Nombre d'heures étudiées}$$

$$\text{Score Prédit} = b + a \times \text{Nombre d'heures étudiées}$$

En fait, nous allons commencer par utiliser la **Descente de Gradient** pour trouver l'**Ordonnée à l'origine (b)**.

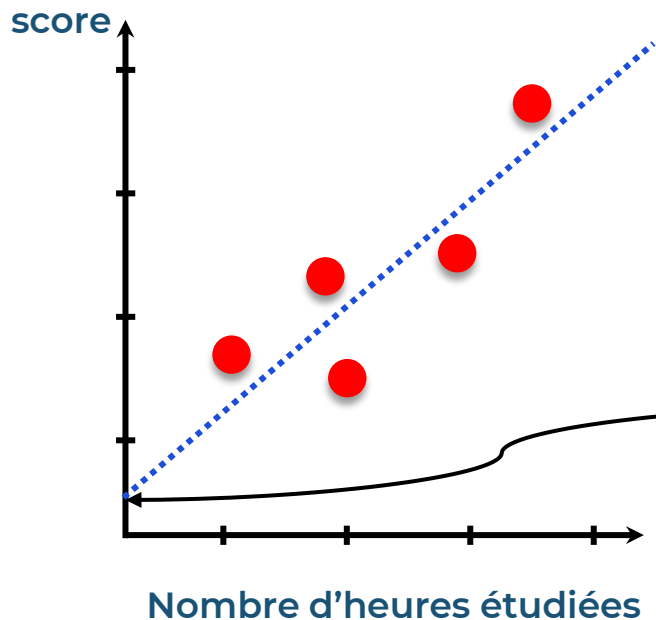
Ensuite, une fois que nous comprendrons comment fonctionne la **Descente de Gradient**, nous l'utiliserons pour résoudre l'**Ordonnée à l'origine (b)** et la **Pente (a)**.



Descente de gradient

Score Prédit = ordonnée à l'origine + $2 \times$ Nombre d'heures étudiées

Score Prédit = $b + 2 \times$ Nombre d'heures étudiées



Pour l'instant, contentons-nous de substituer l'estimation des Moindres Carrés pour la **Pente (a)**,

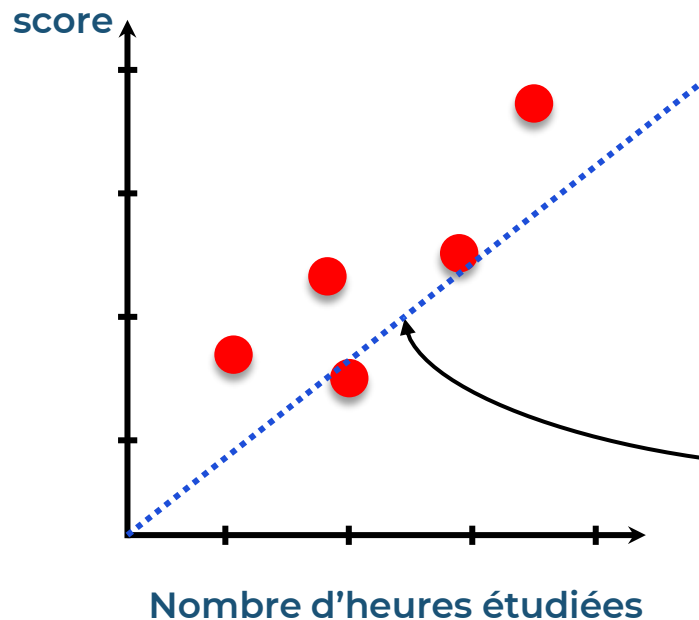
2 (une valeur aléatoire)

...et nous utiliserons la Descente de Gradient pour trouver la valeur optimale pour l'**Ordonnée à l'origine (b)**.

Descente de gradient

$$\text{Score Prédit} = \text{ordonnée à l'origine} + 2 \times \text{Nombre d'heures étudiées}$$

$$\text{Score Prédit} = b + 2 \times \text{Nombre d'heures étudiées}$$



La première chose que nous faisons est de choisir une valeur aléatoire pour l'**Ordonnée à l'origine (b)**.

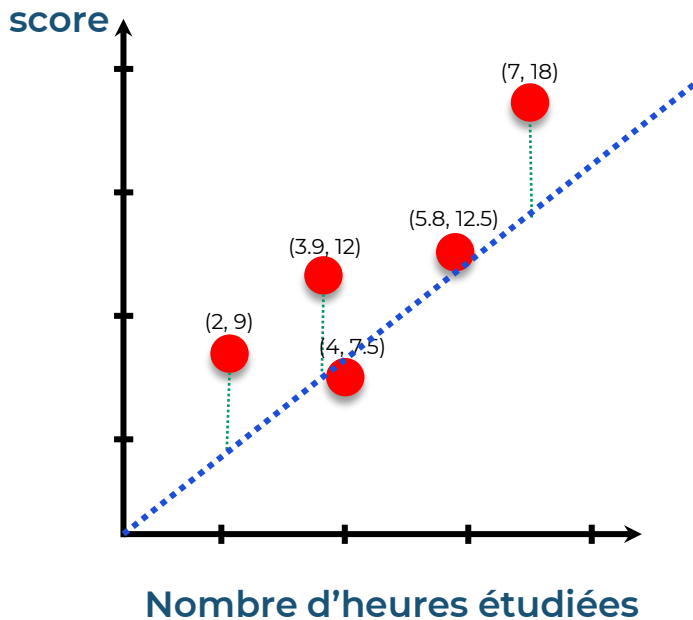
→ Ce n'est qu'une estimation initiale qui donne à la **Descente de Gradient** quelque chose à améliorer.

$$\text{Score Prédit} = 0 + 2 \times \text{Nombre d'heures étudiées}$$

Et cela nous donne l'équation de cette droite.

Descente de gradient

$$\text{Score Prédit} = 0 + 2 \times \text{Nombre d'heures étudiées}$$



Dans cet exemple, nous évaluerons à quel point cette droite s'ajuste aux données avec la **Somme des Résidus au Carré**.

Le résidu est la différence entre le Score Observé et le Score Prédit

$$\text{Résidu} = \text{Score Observé} - \text{Score Prédit}$$

Nombre d'heures étudiées	2	3.9	4	5.8	7
Score (Note) Observé	9	12	7.5	12.5	18
Score (Note) Prédit	4	7.8	8	11.6	14
Résidu	5	4.2	-0.5	0.9	4

$$\text{Somme des résidus au carré} = 5^2 + 4.5^2 + -0.5^2 + 0.9^2 + 4^2 = 59.7$$

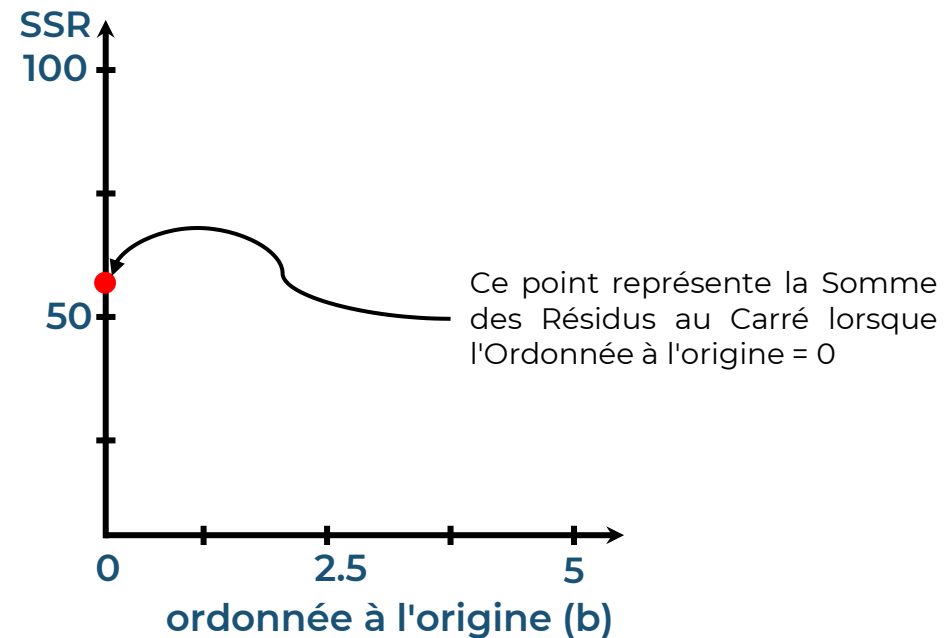
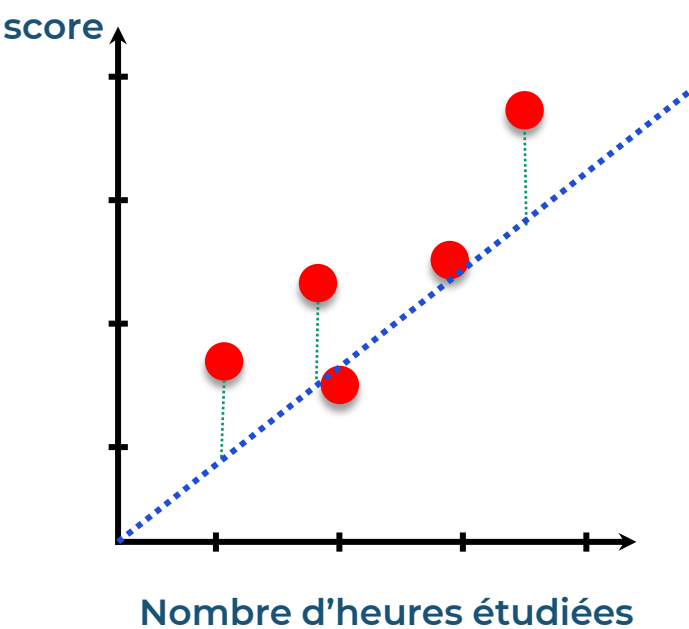
NOTE : Dans le jargon du Machine Learning, la Somme des Résidus au Carré est un type de **Fonction de Coût**.

Descente de gradient

$$\text{Somme des résidus au carré} = 5^2 + 4.5^2 + -0.5^2 + 0.9^2 + 4^2 = 59.7$$

Au final, **59.7** est la Somme des Résidus au Carré.

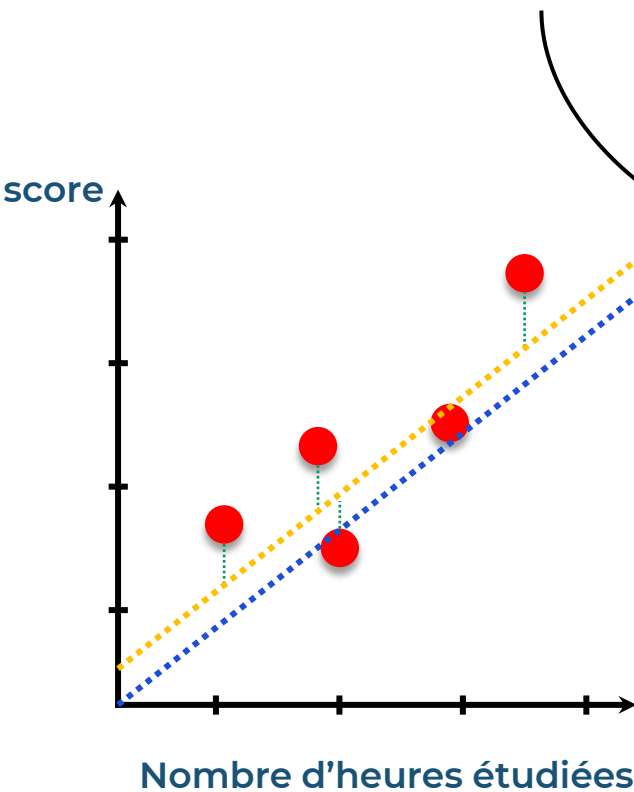
Maintenant, pour s'amuser, nous pouvons tracer cette valeur sur un graphique (SSR vs b).



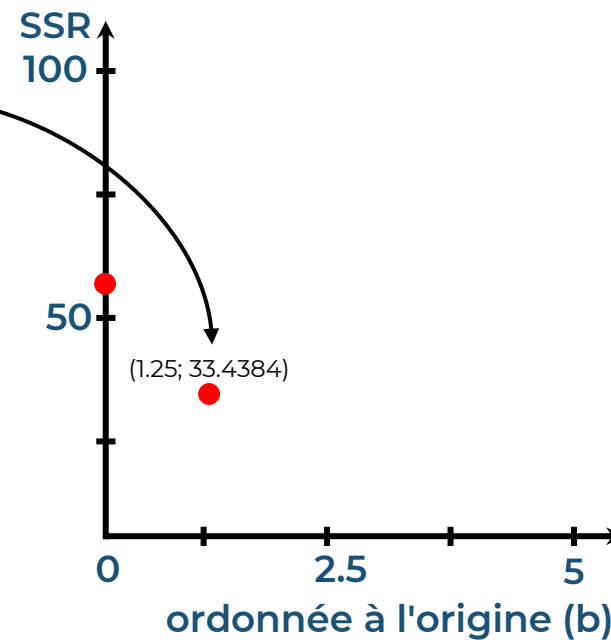
Descente de gradient

Si l'ordonnée à l'origine (b) = 1.25

...alors nous obtiendrions ce point sur le graphique.

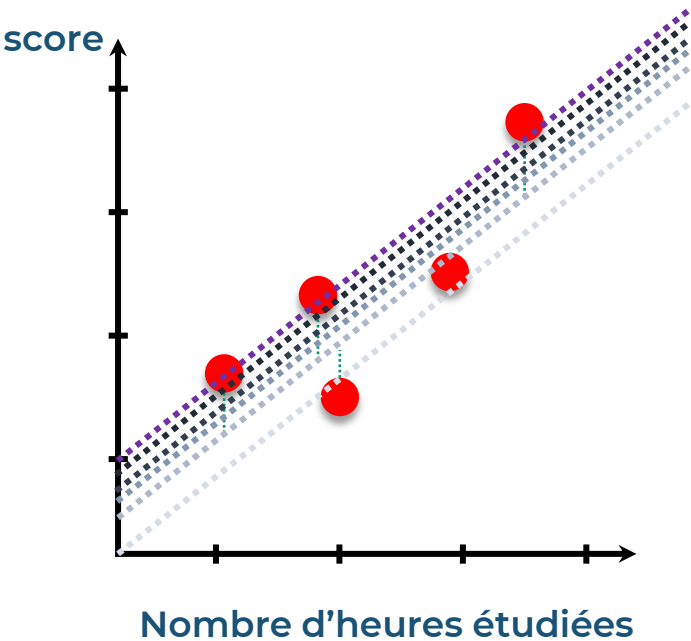


Nombre d'heures étudiées	2	3.9	4	5.8	7
Score (Note) Observé	9	12	7.5	12.5	18
Score (Note) Prédit	5.25	9.05	9.25	12.28	15.25
Résidu	3.75	2.95	-1.75	0.22	2.75

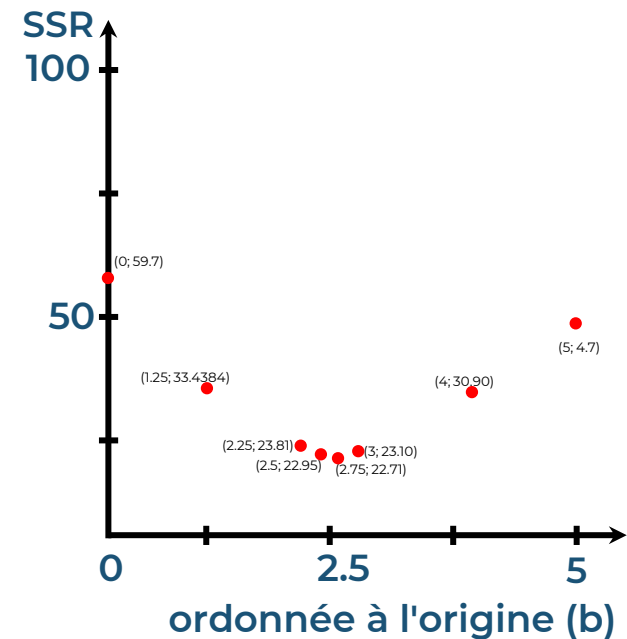


Descente de gradient

Et pour des valeurs l'**Ordonnée à l'origine (b)**, nous obtenons ces points montrés dans le tableau et figure



Nombre d'heures étudiées	2	3.9	4	5.8	7
Score (Note) Observé	9	12	7.5	12.5	18
Score (Note) Prédit (b) = 0	4	7.8	8	11.6	14
Score (Note) Prédit (b) = 1.25	5.25	9.05	9.25	12.28	15.25
Score (Note) Prédit (b) = 2.25	6.25	10.05	10.25	13.85	16.25
Score (Note) Prédit (b) = 2.5	6.5	10.3	10.5	14.1	16.5
Score (Note) Prédit (b) = 2.75	6.75	10.55	10.75	14.35	16.75
Score (Note) Prédit (b) = 3	7	10.8	11	14.6	17
Score (Note) Prédit (b) = 4	8	11.8	12	15.6	18

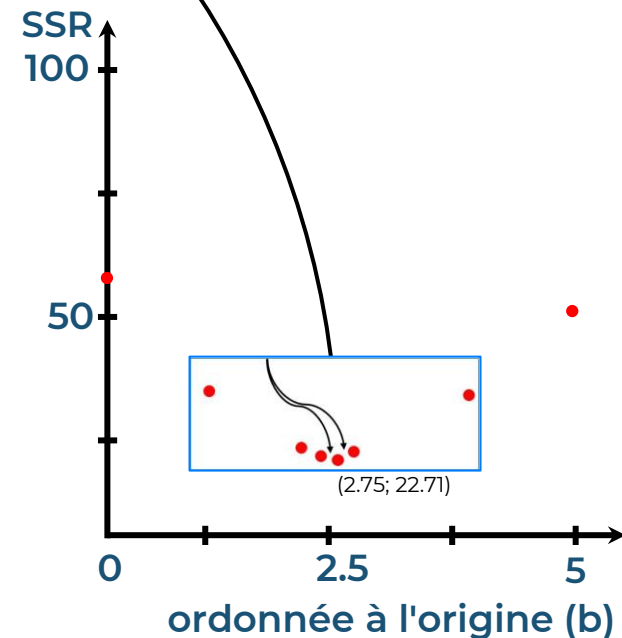


Descente de gradient

Parmi les points que nous avons calculés pour le graphique, celui-ci a la plus faible Somme des Résidus au Carré...

...mais est-ce le meilleur que nous puissions faire ?

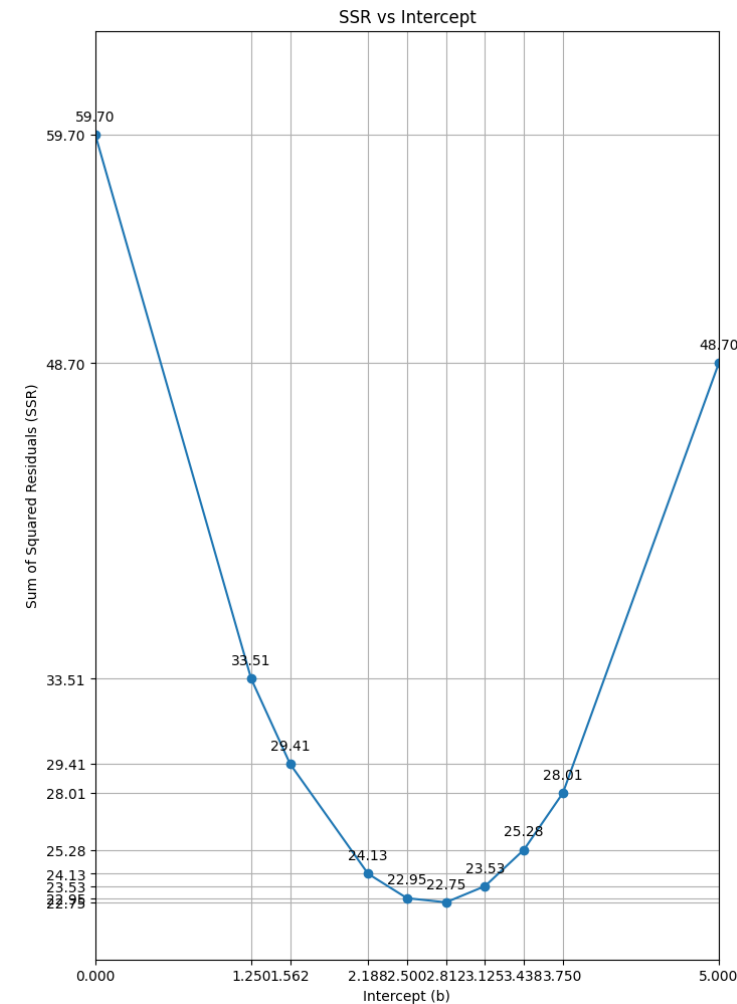
Et si la meilleure valeur pour l'Ordonnée à l'origine se trouvait quelque part entre ces valeurs ?



Descente de gradient

Une méthode lente et fastidieuse pour trouver la Somme minimale des Résidus au Carré consiste à tester à la main un tas d'autres valeurs pour l'**Ordonnée à l'origine**.

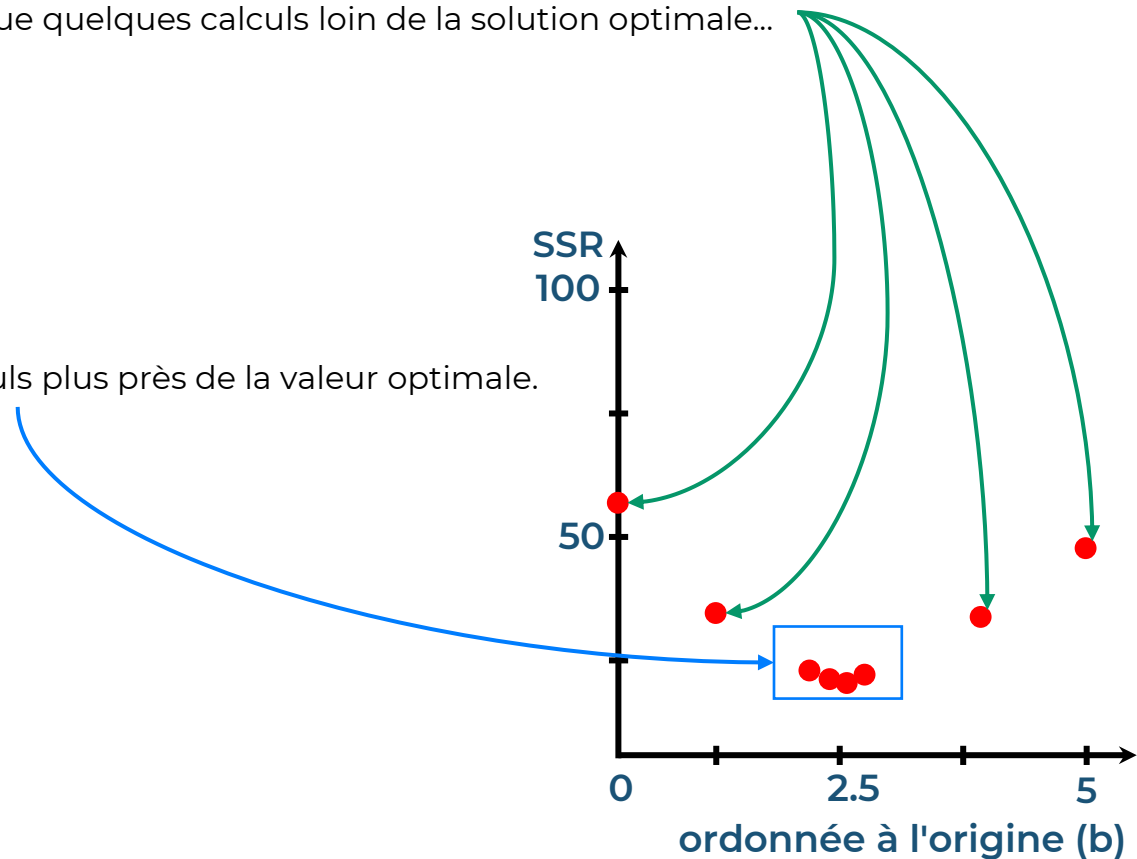
La **Descente de Gradient**
est bien plus efficace !



Descente de gradient

La Descente de Gradient ne fait que quelques calculs loin de la solution optimale...

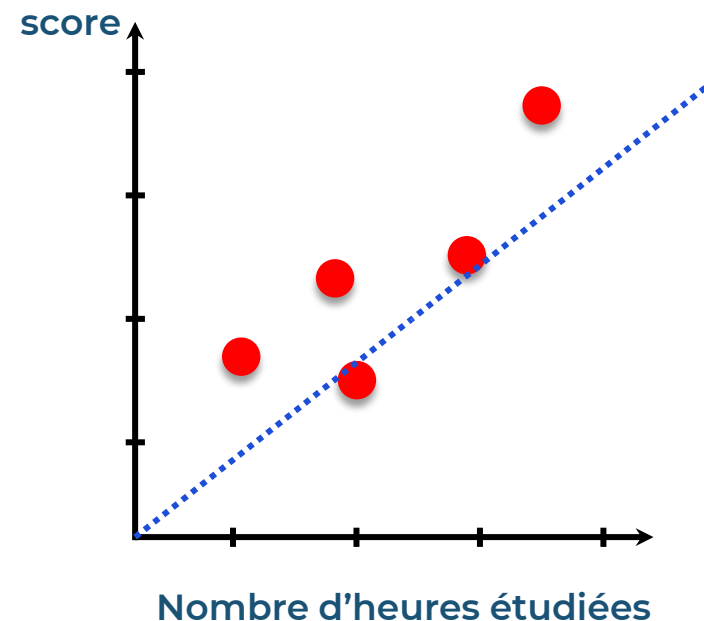
...et augmente le nombre de calculs plus près de la valeur optimale.



Descente de gradient

Revenons donc à l'utilisation de la **Descente de Gradient** pour trouver la valeur optimale de l'**Ordonnée à l'origine (b)**, en partant d'une valeur aléatoire.

Dans ce cas, la valeur aléatoire de **(a)** était **2**.

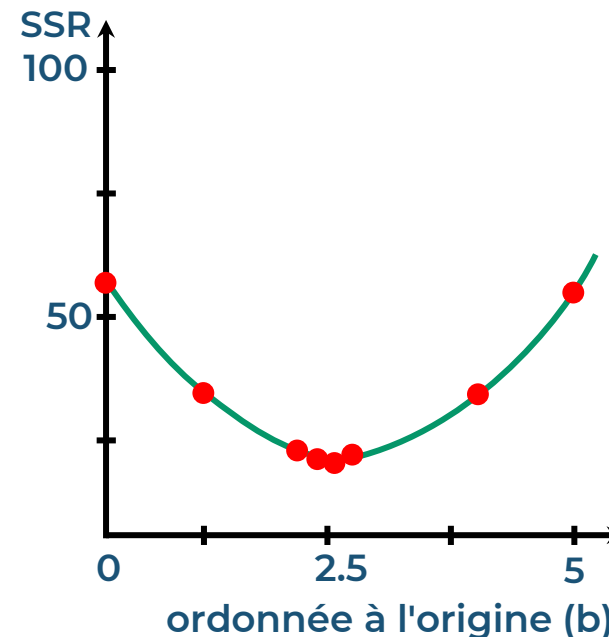


Nombre d'heures étudiées	2	3.9	4	5.8	7
Score (Note) Observé	9	12	7.5	12.5	18
Score (Note) Prédit	4	7.8	8	11.6	14
Résidu	5	4.2	-0.5	0.9	4

$$SSR = (9 - (2 * 2 + b))^2 + (12 - (2 * 3.9 + b))^2 + (7.5 - (2 * 4 + b))^2 + (12.5 - (2 * 5.8 + b))^2 + (18 - (2 * 7 + b))^2$$

Maintenant, nous pouvons facilement substituer n'importe quelle valeur pour l'**ordonnée à l'origine (Intercept - b)** ...

...et obtenir la **Somme des Résidus au Carré**

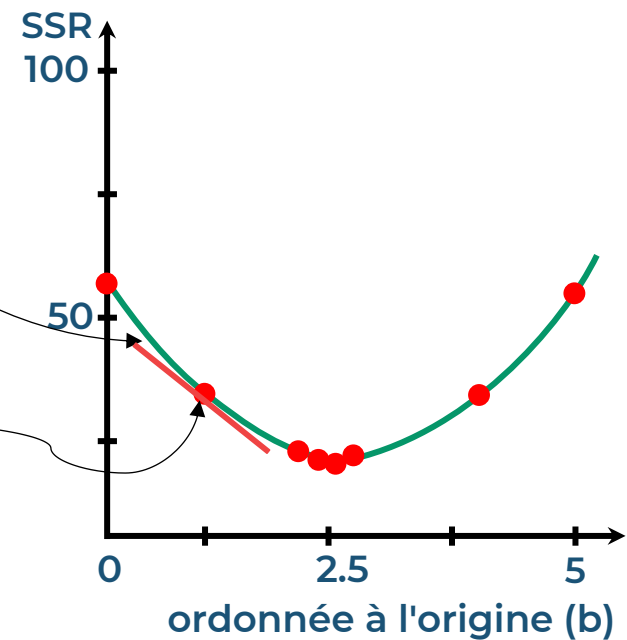


Descente de gradient

$$SSR = (9 - (2 * 2 + b))^2 + (12 - (2 * 3.9 + b))^2 + (7.5 - (2 * 4 + b))^2 + (12.5 - (2 * 5.8 + b))^2 + (18 - (2 * 7 + b))^2$$

Ainsi, nous avons maintenant une équation pour cette courbe...

...et nous pouvons prendre la dérivée de cette fonction et déterminer **la pente** pour toute valeur de l'**Ordonnée à l'origine**.

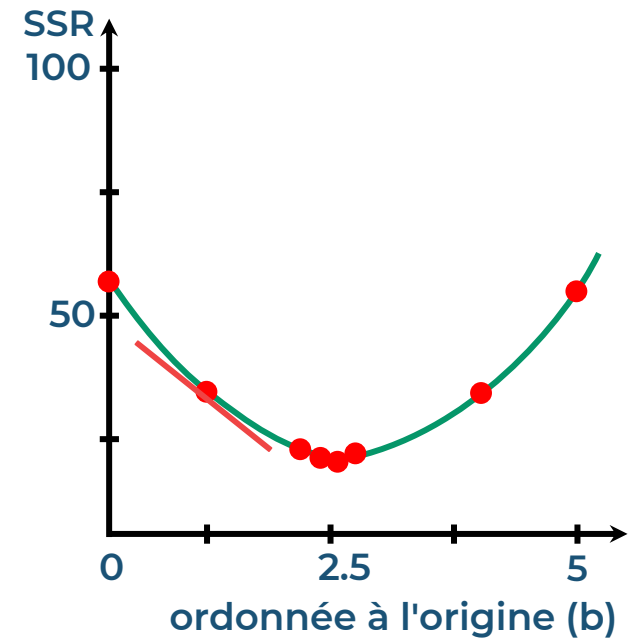


Descente de gradient

$$SSR = (9 - (2 * 2 + b))^2 + (12 - (2 * 3.9 + b))^2 + (7.5 - (2 * 4 + b))^2 + (12.5 - (2 * 5.8 + b))^2 + (18 - (2 * 7 + b))^2$$

Prenons donc la dérivée de la Somme des Résidus au Carré par rapport à l'**Ordonnée à l'origine (b)**.

$$\begin{aligned} \frac{\delta}{\delta b} SSR &= \frac{\delta}{\delta b} (9 - (2 * 2 + b))^2 \\ &+ \frac{\delta}{\delta b} (12 - (2 * 3.9 + b))^2 \\ &+ \frac{\delta}{\delta b} (7.5 - (2 * 4 + b))^2 \\ &+ \frac{\delta}{\delta b} (12.5 - (2 * 5.8 + b))^2 \\ &+ \frac{\delta}{\delta b} (18 - (2 * 7 + b))^2 \end{aligned}$$



$$(U + V)' = U' + V'$$

Descente de gradient

...la dérivée de la première partie...

$$\frac{\delta}{\delta b} (9 - (2 * 2 + b))^2 = -2(9 - (2 * 2 + b))$$

...De même pour le reste ...

$$\frac{\delta}{\delta b} (12 - (2 * 3.9 + b))^2 = -2(12 - (2 * 3.9 + b))$$

$$\frac{\delta}{\delta b} (7.5 - (2 * 4 + b))^2 = -2(7.5 - (2 * 4 + b))$$

$$\frac{\delta}{\delta b} (12.5 - (2 * 5.8 + b))^2 = -2(12.5 - (2 * 5.8 + b))$$

$$\frac{\delta}{\delta b} (18 - (2 * 7 + b))^2 = -2(18 - (2 * 7 + b))$$

$$(U^2)' = 2U'U$$

Descente de gradient

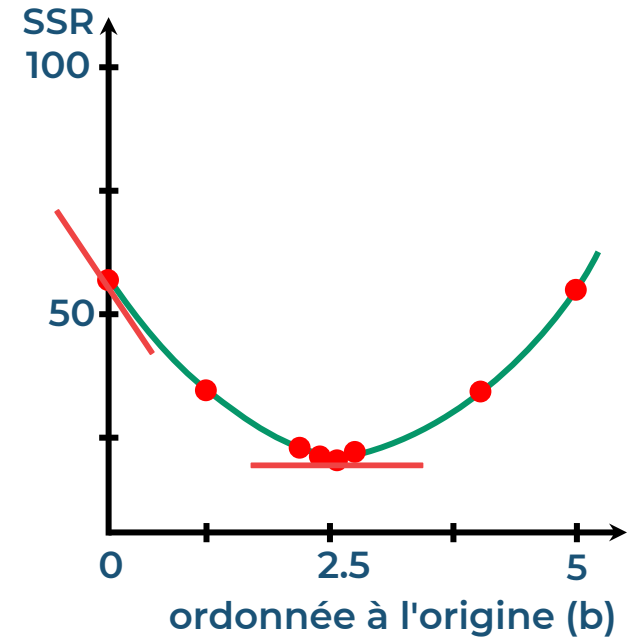
$$\begin{aligned}\frac{\delta}{\delta b} SSR &= -2(9 - (2 * 2 + b)) \\ &+ -2(12 - (2 * 3.9 + b)) \\ &+ -2(7.5 - (2 * 4 + b)) \\ &+ -2(12.5 - (2 * 5.8 + b)) \\ &+ -2(18 - (2 * 7 + b))\end{aligned}$$

Maintenant que nous avons la dérivée, la **Descente de Gradient** l'utilisera pour trouver où la Somme des Résidus au Carré est la plus faible.

$$\begin{aligned}\frac{\delta}{\delta b} SSR &= -2(9 - (2 * 2 + 0)) \\ &+ -2(12 - (2 * 3.9 + 0)) \\ &+ -2(7.5 - (2 * 4 + 0)) \\ &+ -2(12.5 - (2 * 5.8 + 0)) \\ &+ -2(18 - (2 * 7 + 0)) = -27.2\end{aligned}$$

Donc, lorsque l'**Ordonnée à l'origine (b)**= 0, la pente de la courbe = -27.2

NOTE : Plus nous nous rapprochons de la valeur optimale pour l'**Ordonnée à l'origine**, plus la pente de la courbe se rapproche de 0.

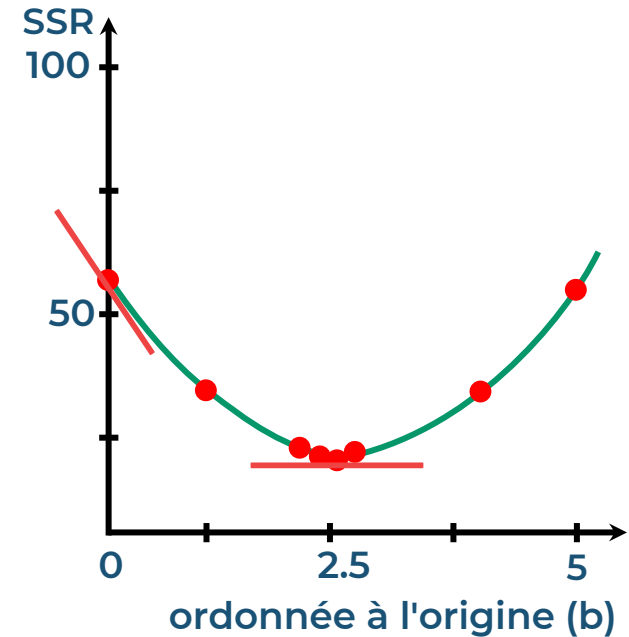


Descente de gradient

.Quand la pente est loin de 0...

... nous devrions faire de grands pas, car nous sommes loin de la valeur optimale.

Et nous devrions faire de petits pas, lorsque nous sommes proches de la valeur optimale...



Taille du Pas = Pente x Taux d'Apprentissage

$$\text{Taille du Pas} = -27.2 \times 0.1 = -2.72$$

la taille du pas doit être liée à la pente, puisqu'elle nous indique si nous devons faire un petit pas ou un grand pas, mais nous devons nous assurer que le grand pas n'est pas trop grand.

Descente de gradient

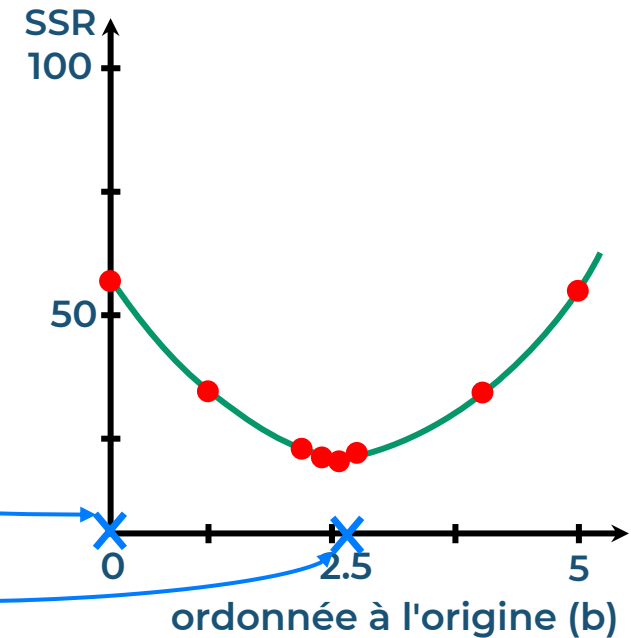
$$\text{Taille du Pas} = -27.2 \times 0.1 = -2.72$$

$$\text{Nouvelle (b)} = \text{Ancienne (b)} - \text{Taille du Pas}$$

$$= 0 - (-2.72)$$

$$= 2.72$$

...et la Nouvelle Ordonnée à l'origine = 2.72.



Descente de gradient

la Nouvelle Ordonnée à l'origine = 2.72.

$$\begin{aligned} \frac{\delta}{\delta b} SSR &= -2(9 - (2 * 2 + 2.72)) \\ &+ -2(12 - (2 * 3.9 + 2.72)) \\ &+ -2(7.5 - (2 * 4 + 2.72)) \\ &+ -2(12.5 - (2 * 5.8 + 2.72)) \\ &+ -2(18 - (2 * 7 + 2.72)) = -13.68 \end{aligned}$$

Pour faire un autre pas, nous revenons à la dérivée et substituons

la **Nouvelle Ordonnée à l'origine** (2.72)...

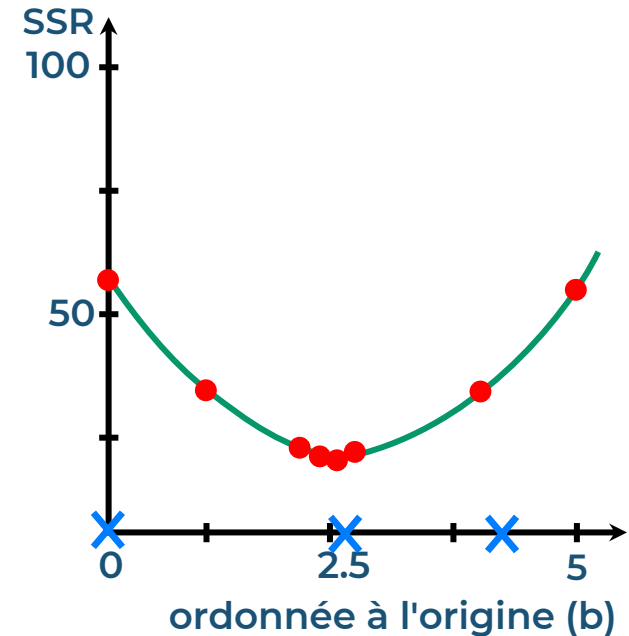
...et cela nous dit que la pente de la courbe = -13.68

$$\text{Taille du Pas} = -13.68 \times 0.1 = -1.368$$

$$\text{Nouvelle (b)} = \text{Ancienne (b)} - \text{Taille du Pas}$$

$$= 2.72 - (-1.368)$$

$$= 4.088$$



Descente de gradient

la Nouvelle Ordonnée à l'origine = 4.088.

$$\begin{aligned} \frac{\delta}{\delta b} SSR &= -2(9 - (2 * 2 + 4.088)) \\ &+ -2(12 - (2 * 3.9 + 4.088)) \\ &+ -2(7.5 - (2 * 4 + 4.088)) \\ &+ -2(12.5 - (2 * 5.8 + 4.088)) \\ &+ -2(18 - (2 * 7 + 4.088)) = 13.68 \end{aligned}$$

Pour faire un autre pas, nous revenons à la dérivée et substituons

la **Nouvelle Ordonnée à l'origine** (4.088)...

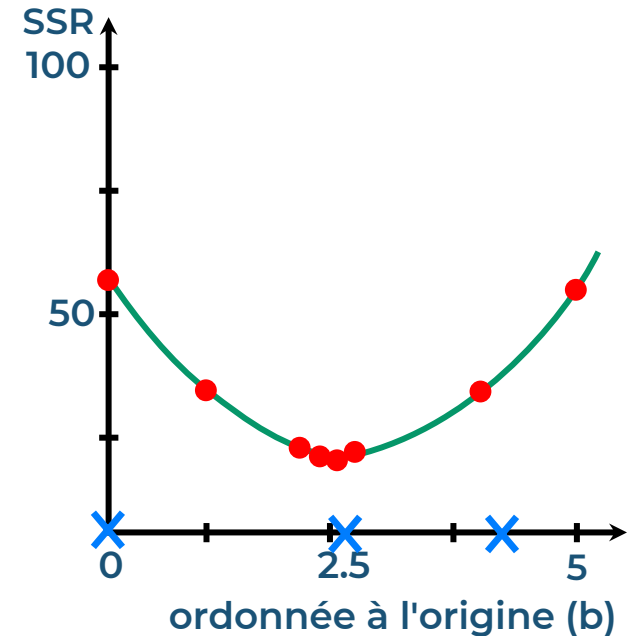
...et cela nous dit que la pente de la courbe = 13.68

$$\text{Taille du Pas} = 13.68 \times 0.1 = 1.368$$

$$\text{Nouvelle (b)} = \text{Ancienne (b)} - \text{Taille du Pas}$$

$$= 4.088 - (1.368)$$

$$= 2.72$$

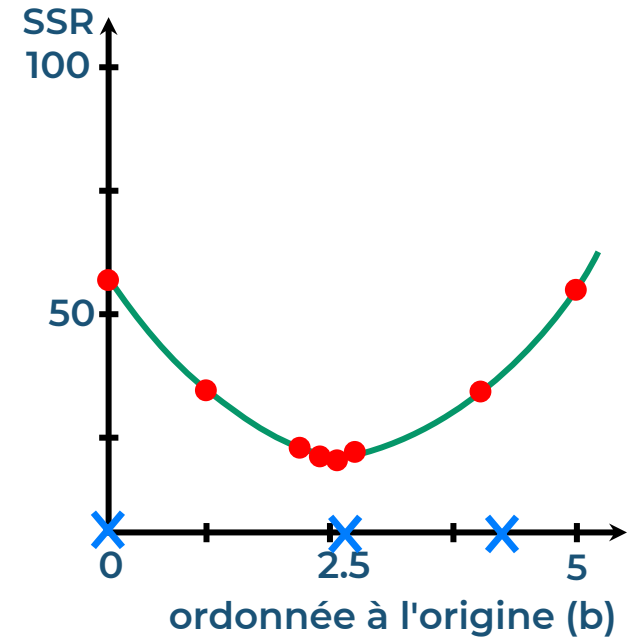


Descente de gradient

la Nouvelle Ordonnée à l'origine = 2.72.

$$\begin{aligned} \frac{\partial}{\partial b} SSR &= -2(9 - (2 * 2 + 2.72)) \\ &+ -2(12 - (2 * 3.9 + 2.72)) \\ &+ -2(7.5 - (2 * 4 + 2.72)) \\ &+ -2(12.5 - (2 * 5.8 + 2.72)) \\ &+ -2(18 - (2 * 7 + 2.72)) = 0 \end{aligned}$$

Pour faire un autre pas, nous revenons à la dérivée et substituons la **Nouvelle Ordonnée à l'origine** (2.72)



La **Descente de Gradient** s'arrête lorsque la **Taille du Pas** est **Très Proche de 0**.

La Taille du Pas sera Très Proche de 0 lorsque la Pente est très proche de 0.

Descente de gradient

En pratique, la Taille du Pas Minimum = 0.001
ou plus petit.

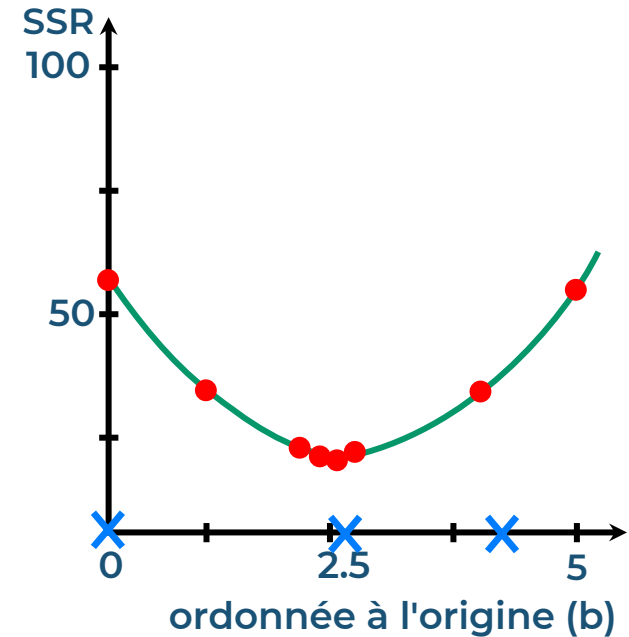
Taille du Pas = Pente × Taux d'Apprentissage

...et obtenir 0.0009, qui est inférieur à 0.001, donc la Descente de Gradient s'arrêterait.

Taille du Pas = $0.009 \times 0.1 = 0.0009$

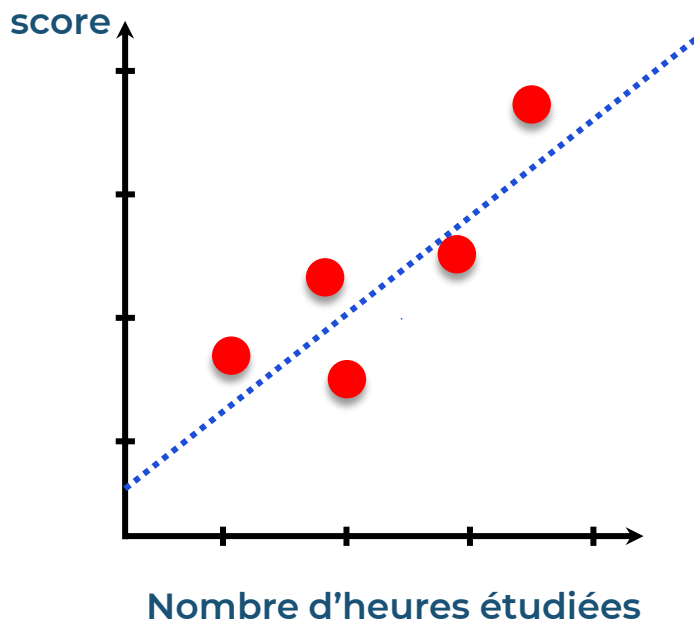
Cela dit, la **Descente de Gradient** inclut également une limite sur le nombre de pas qu'elle effectuera avant d'abandonner.

En pratique, le **Nombre Maximum de Pas = 1 000** ou plus.



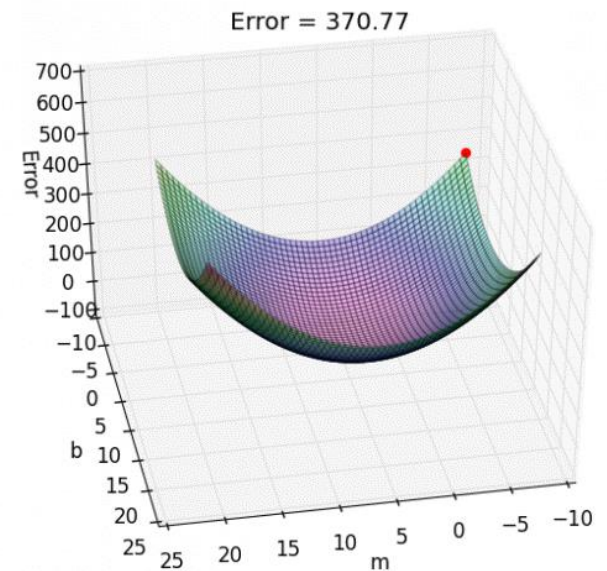
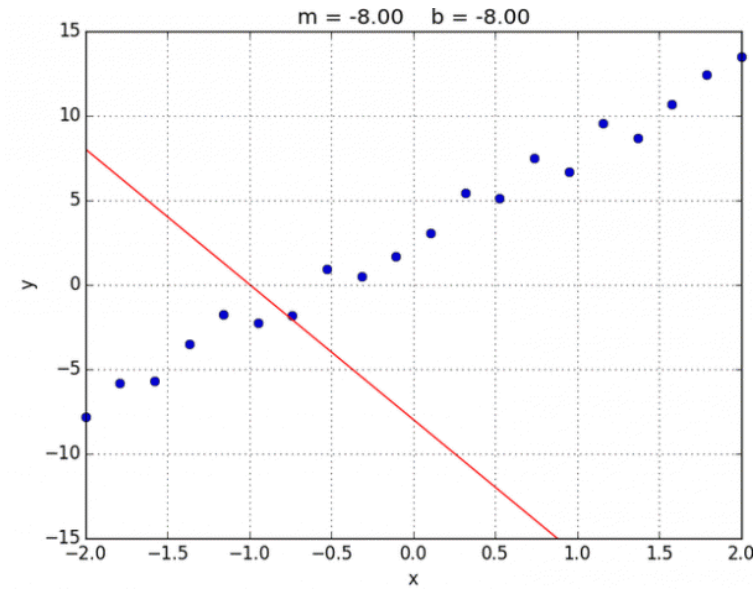
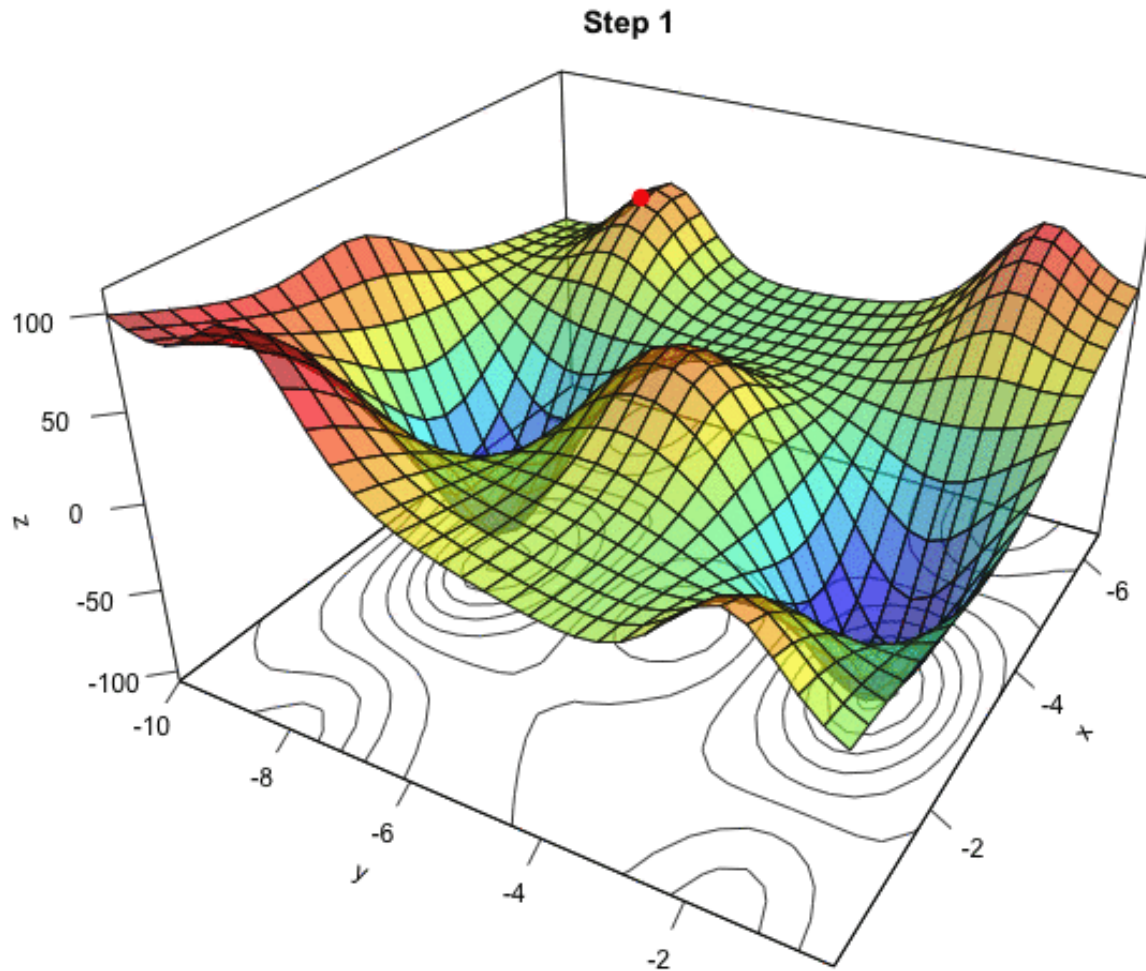
Descente de gradient

Score Prédit = ordonnée à l'origine + pente × **Nombre d'études**



...parlons de la façon d'estimer l'**Ordonnée à l'origine** et la **Pente**.

Descente de gradient



Descente de gradient

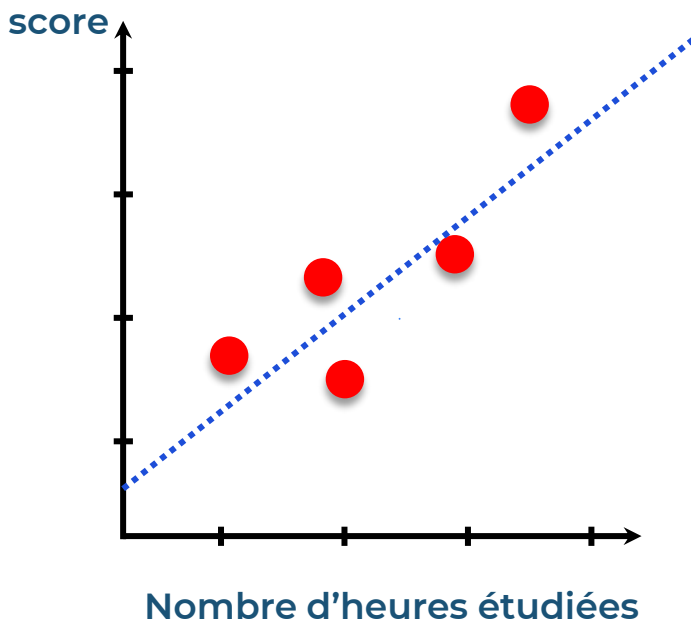
$$SSR = (9 - (2 * 2 + b))^2 + (12 - (2 * 3.9 + b))^2 + (7.5 - (2 * 4 + b))^2 + (12.5 - (2 * 5.8 + b))^2 + (18 - (2 * 7 + b))^2$$

Comme avant, nous utiliserons la Somme des Résidus au Carré comme **Fonction de Coût**

...et comme avant, nous prendrons la dérivée par rapport à l'**Ordonnée à l'origine...Pente!**

$$\frac{\delta}{\delta Pente} SSR$$

$$\frac{\delta}{\delta Ordonnée à l'origine} SSR$$



Descente de gradient

$$\begin{aligned}\frac{\delta}{\delta b} SSR &= -2(9 - (a * 2 + b)) \\ &+ -2(12 - (a * 3.9 + b)) \\ &+ -2(7.5 - (a * 4 + b)) \\ &+ -2(12.5 - (a * 5.8 + b)) \\ &+ -2(18 - (a * 7 + b))\end{aligned}$$

Voici la dérivée de la Somme des Résidus au Carré par rapport à l'**Ordonnée à l'origine...**

$$\begin{aligned}\frac{\delta}{\delta a} SSR &= -2 \times 2(9 - (a * 2 + b)) \\ &+ -2 \times 3.9(12 - (a * 3.9 + b)) \\ &+ -2 \times 4(7.5 - (a * 4 + b)) \\ &+ -2 \times 5.8(12.5 - (a * 5.8 + b)) \\ &+ -2 \times 7(18 - (a * 7 + b))\end{aligned}$$

...et voici la dérivée par rapport à la Pente.

NOTE : Lorsque vous avez deux dérivées ou plus de la même fonction, elles sont appelées un **Gradient**.

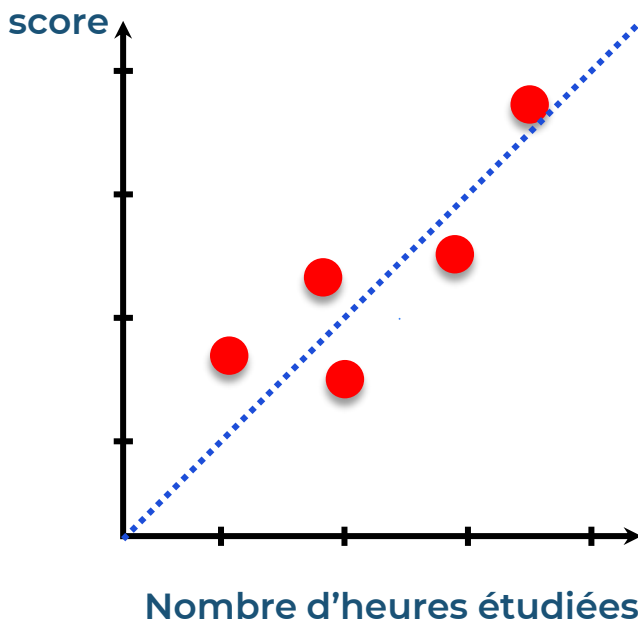
Nous utiliserons ce Gradient pour descendre au point le plus bas de la Fonction de Coût, qui, dans ce cas, est la Somme des Résidus au Carré...

...c'est pourquoi cet algorithme s'appelle la **Descente de Gradient !**

Descente de gradient

Comme avant, nous commencerons par choisir un nombre aléatoire pour l'**Ordonnée à l'origine**. Dans ce cas, nous définirons l'**Ordonnée à l'origine** = 0...

...et nous choisirons un nombre aléatoire pour la **Pente**. Dans ce cas, nous définirons la **Pente** = 2.



$$\begin{aligned} \frac{\partial}{\partial b} SSR &= -2(9 - (2 * 2 + 0)) \\ &+ -2(12 - (2 * 3.9 + 0)) \\ &+ -2(7.5 - (2 * 4 + 0)) \\ &+ -2(12.5 - (2 * 5.8 + 0)) \\ &+ -2(18 - (2 * 7 + 0)) = -27.2 \end{aligned}$$

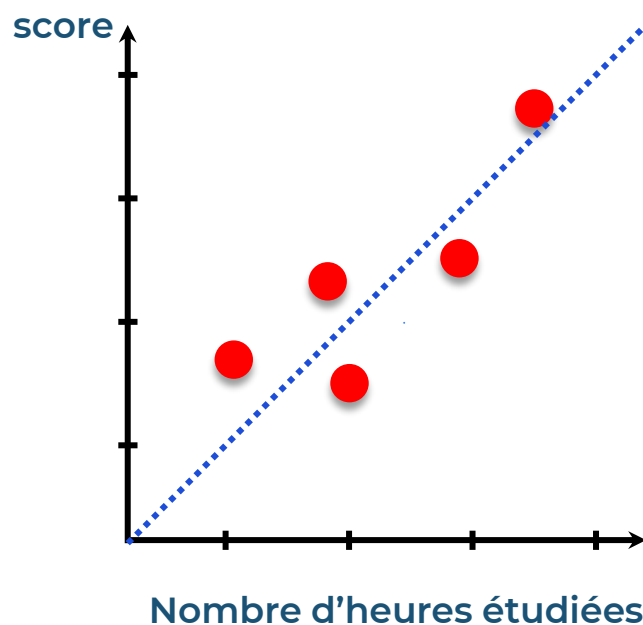
$$\begin{aligned} \frac{\partial}{\partial a} SSR &= -2 \times 2(9 - (2 * 2 + 0)) \\ &+ -2 \times 3.9(12 - (2 * 3.9 + 0)) \\ &+ -2 \times 4(7.5 - (2 * 4 + 0)) \\ &+ -2 \times 5.8(12.5 - (2 * 5.8 + 0)) \\ &+ -2 \times 7(18 - (2 * 7 + 0)) = -115.2 \end{aligned}$$

Descente de gradient

...maintenant, nous introduisons les Pentes dans les formules de Taille du Pas...

$$\begin{aligned} \frac{\delta}{\delta b} SSR &= -2(9 - (2 * 2 + 0)) \\ &+ -2(12 - (2 * 3.9 + 0)) \\ &+ -2(7.5 - (2 * 4 + 0)) \\ &+ -2(12.5 - (2 * 5.8 + 0)) \\ &+ -2(18 - (2 * 7 + 0)) = -27.2 \end{aligned}$$

$$\begin{aligned} \frac{\delta}{\delta a} SSR &= -2 \times 2(9 - (2 * 2 + 0)) \\ &+ -2 \times 3.9(12 - (2 * 3.9 + 0)) \\ &+ -2 \times 4(7.5 - (2 * 4 + 0)) \\ &+ -2 \times 5.8(12.5 - (2 * 5.8 + 0)) \\ &+ -2 \times 7(18 - (2 * 7 + 0)) = -115.2 \end{aligned}$$



$$\begin{aligned} \text{Taille du Pas Intercept}(b) &= -27.2 \times \text{Taux d'Apprentissage} \\ &= -27.2 * 0.01 \\ &= -0.272 \end{aligned}$$

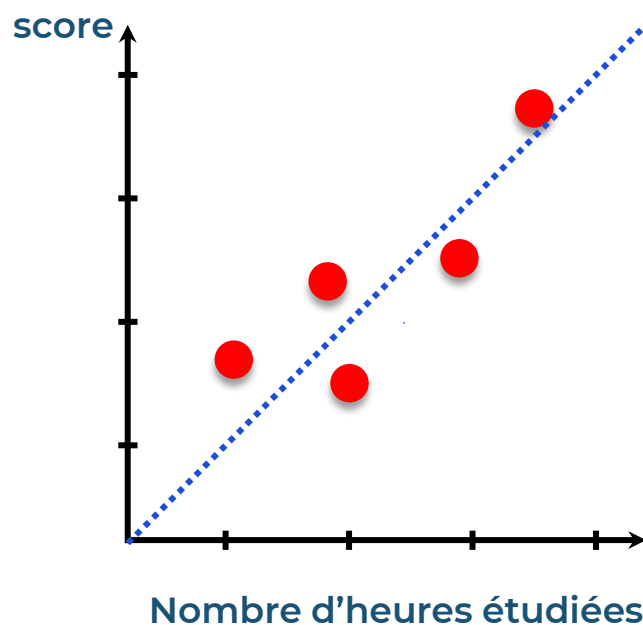
$$\begin{aligned} \text{Taille du Pas Pente}(a) &= -115.2 \times \text{Taux d'Apprentissage} \\ &= -115.2 * 0.01 \\ &= -1.152 \end{aligned}$$

Descente de gradient

Maintenant, nous calculons la Nouvelle Ordonnée à l'origine et la Nouvelle Pente en substituant l'Ancienne Ordonnée à l'origine et l'Ancienne Pente...

$$\begin{aligned} \frac{\delta}{\delta b} SSR &= -2(9 - (2 * 2 + 0)) \\ &+ -2(12 - (2 * 3.9 + 0)) \\ &+ -2(7.5 - (2 * 4 + 0)) \\ &+ -2(12.5 - (2 * 5.8 + 0)) \\ &+ -2(18 - (2 * 7 + 0)) = -27.2 \end{aligned}$$

$$\begin{aligned} \frac{\delta}{\delta a} SSR &= -2 \times 2(9 - (2 * 2 + 0)) \\ &+ -2 \times 3.9(12 - (2 * 3.9 + 0)) \\ &+ -2 \times 4(7.5 - (2 * 4 + 0)) \\ &+ -2 \times 5.8(12.5 - (2 * 5.8 + 0)) \\ &+ -2 \times 7(18 - (2 * 7 + 0)) = -115.2 \end{aligned}$$

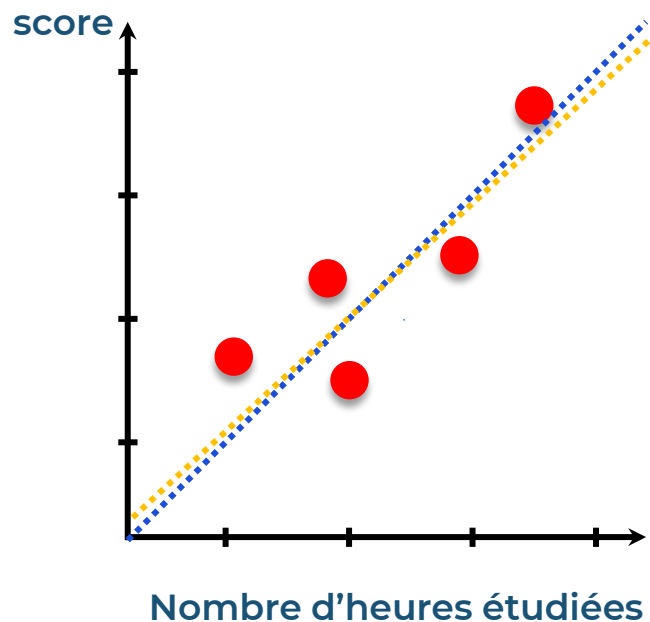


$$\begin{aligned} \text{Nouvelle intercept (b)} &= \text{Ancienne intercept (b)} - \text{Taille du Pas} \times \text{Intercept (b)} \\ &= 0 - 0.272 \\ &= 0.272 \end{aligned}$$

$$\begin{aligned} \text{Nouvelle Pente(a)} &= \text{Ancienne Pente(a)} - \text{Taille du Pas} \times \text{Pente(a)} \\ &= 2 - 1.152 \\ &= 3.152 \end{aligned}$$

Descente de gradient

...et voici la nouvelle droite (avec Pente = 3.152 et Ordonnée à l'origine = 0.272) après la première étape.



Maintenant, nous répétons simplement ce que nous avons fait jusqu'à ce que toutes les **Tailles de Pas** soient très petites ou que nous atteignons le **Nombre Maximum de Pas**.

La droite qui ajuste le mieux les données, a une Ordonnée à l'origine = 4.115 et Pente=1.7, les mêmes valeurs que nous obtenons des Moindres Carrés.

Descente de gradient

- **Étape 1** : Prendre la dérivée de la **Fonction de Coût** pour chaque paramètre qu'elle contient. En jargon Machine Learning, prendre le **Gradient** de la **Fonction de Coût**.
- **Étape 2** : Choisir des valeurs aléatoires pour les paramètres.
- **Étape 3** : Substituer les valeurs des paramètres dans les dérivées (c'est-à-dire, le **Gradient**).
- **Étape 4** : Calculer les Tailles de Pas :

$$\text{Taille du Pas} = \text{résultat étape 3} \times \text{Taux d'Apprentissage}$$

- **Étape 5** : Calculer les Nouveaux Paramètres :

$$\text{Nouveau Paramètre} = \text{Ancien Paramètre} - \text{Taille du Pas}$$

L'algo consiste à retourner à l'**Étape 3** et répéter jusqu'à ce que la **Taille du Pas** soit très petite, ou qu'il atteigne le **Nombre Maximum de Pas**.

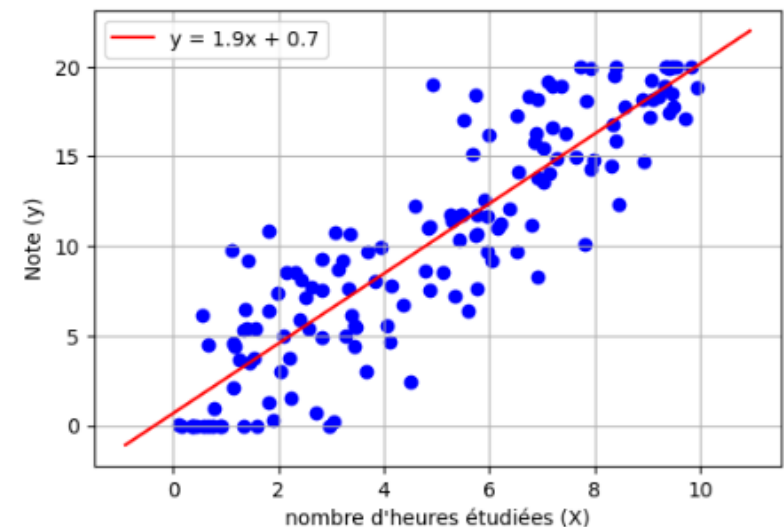
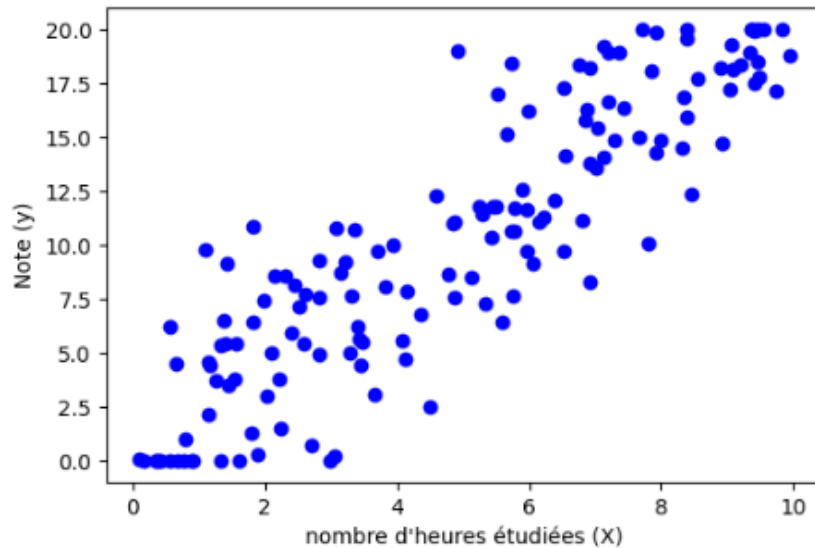
Descente de gradient

Activité:

Réalisez un script Python pour implémenter l'algorithme de la descente de gradient en définissant les 4 étapes fondamentales :

1. dataset(X,y),
2. le modèle, - initialisez les paramètres par des valeurs aléatoire (ex. $a = 1$ et $b = 0$) pour faire des premières prédictions
3. fonction coût
4. et algorithme d'optimisation (gradient) – condition d'arrêt nombre d'itération = 1000

Utiliser le dataset suivant : [score_hourse.csv](#) (contient 150 lignes des heures d'études et les notes correspondantes, aléatoires avec corrélation forte $=0.9$)



IA & Machine Learning (M354)



Ch4. Apprentissage Supervisé - Régression linéaire (Suite)



Notation Matriciel



Régression Linéaire Polynomiale – cas pratique



Régression Linéaire Multiple



La régression avec la bibliothèque Sckit-learn

Notation Matriciel

Dataset

↪ (x, y) avec m exemples
et n variables ($n=1$)

x	y
$x^{(1)}$	$y^{(1)}$
....
$x^{(m)}$	$y^{(m)}$

Modèle

$$\hookrightarrow f(x^{(i)}) = ax^{(i)} + b$$

Fonction Coût

$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 = \frac{1}{2m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})^2$$

Algorithme de Minimisation

$$\frac{\delta J(a, b)}{\delta a} = \frac{1}{m} \sum_{i=1}^m x^{(i)} (ax^{(i)} + b - y^{(i)})$$

$$\frac{\delta J(a, b)}{\delta b} = \frac{1}{m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})$$



$$\begin{cases} a = a - \alpha \frac{\delta J(a, b)}{\delta a} \\ b = b - \alpha \frac{\delta J(a, b)}{\delta b} \end{cases}$$

↪ Nous allons les convertir les équations déjà vu en des matrices pour des raisons suivants :

- L'utilisation des matrices simplifier énormément de calculs
- Nous permettons de développer des modèles plus complexes comme la régression polynomiale



C'est important de connaître la forme matricielle de tout ces équations vu que la programmation Python ou avec des Framework AI utilisent toujours les matrices

Notation Matriciel

Dataset

↪ (x, y) avec m exemples
et n variables ($n=1$)

x	y
$x^{(1)}$	$y^{(1)}$
....
$x^{(m)}$	$y^{(m)}$

Modèle

↪ $f(x^{(i)}) = ax^{(i)} + b$

✓ \mathbf{X} : Matrice de taille $m \times (n + 1)$ ici $n = 1$, donc $m \times 2$

✓ Le modèle prédit \hat{y} (noté F) comme une combinaison linéaire de X et des paramètres θ

$$F = X \cdot \theta$$

$$\begin{bmatrix} f(x^{(1)}) \\ f(x^{(2)}) \\ \vdots \\ f(x^{(m)}) \end{bmatrix} = \begin{bmatrix} x^{(1)} & 1 \\ x^{(2)} & 1 \\ \vdots & \vdots \\ x^{(m)} & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix}$$

$m \times 1$ $m \times (n + 1)$ $(n + 1) \times 1$

Notation Matriciel

🎯 Fonction Coût

↳ Ça c'était notre fonction coût

$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 = \frac{1}{2m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})^2$$

$\theta = \begin{bmatrix} a \\ b \end{bmatrix}$
 $X \cdot \theta = \begin{bmatrix} x^{(1)} & 1 \\ x^{(2)} & 1 \\ \vdots & \vdots \\ x^{(m)} & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix}$
 $Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$

$$J(\theta) = \frac{1}{2m} \sum (X\theta - Y)^2$$

$$= \frac{1}{2m} (X\theta - Y)^T (X\theta - Y)$$

L'objectif est de minimiser l'erreur quadratique moyenne entre les prédictions F et les vraies valeurs y

↳ **NB.** La fonction coût est toujours un scalaire, c'est la moyenne de toutes nos erreurs, c'est-à-dire on ne cherche pas à l'exprimer sous format d'un vecteur ou une matrice.



$(X\theta - Y)^T (X\theta - Y)$ calcule la somme des carrés des résidus de manière vectorielle.

DÉMONSTRATION DANS LE SLIDE SUIVANT

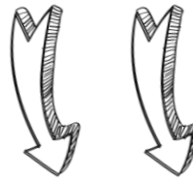
Notation Matriciel

DÉMONSTRATION

$$\sum (X\theta - Y)^2 = (X\theta - Y)^T (X\theta - Y)$$

$$X\theta - Y = \begin{bmatrix} (x^{(1)}\theta - y^{(1)}) \\ (x^{(2)}\theta - y^{(2)}) \\ \dots \\ (x^{(m)}\theta - y^{(m)}) \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_m \end{bmatrix}$$

$$(X\theta - Y)^T = [e_1 \quad e_2 \quad \dots \quad e_m]$$



$$(X\theta - Y)^T (X\theta - Y) = [e_1 \quad e_2 \quad \dots \quad e_m] \times \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_m \end{bmatrix} = e_1 \times e_1 + e_2 \times e_2 + \dots + e_m \times e_m = \sum_{i=1}^m e_i^2$$

Notation Matriciel



Algorithme de Minimisation

MÉTHODE DES MOINDRES CARRÉS

$$\begin{aligned}
 \frac{\delta J(\theta)}{\delta \theta} = 0 & \iff \frac{\delta \left(\frac{1}{2m} (X\theta - Y)^T (X\theta - Y) \right)}{\delta \theta} = 0 \\
 & \iff \frac{1}{2m} \cdot 2 \cdot (X\theta - Y)^T \cdot \frac{\delta (X\theta - Y)}{\delta \theta} = 0 \\
 & \iff \frac{1}{m} (X\theta - Y)^T \cdot X = 0
 \end{aligned}$$

Pour avoir un vecteur colonne :

$$\frac{1}{m} X^T (X\theta - Y) = 0$$

En résolvant, on obtient :

$$\theta = (X^T X)^{-1} X^T Y$$

$$X^T (X\theta - Y) = 0 \Rightarrow X^T X\theta - X^T Y = 0$$

$$\Rightarrow X^T X\theta = X^T Y$$

$$\Rightarrow (X^T X)^{-1} X^T X\theta = (X^T X)^{-1} X^T Y$$

Notation Matriciel



Algorithme de Minimisation

MÉTHODE Gradient Descent

$$\frac{\delta J(a, b)}{\delta a} = \frac{1}{m} \sum_{i=1}^m x^{(i)} (ax^{(i)} + b - y^{(i)})$$

$$\frac{\delta J(a, b)}{\delta b} = \frac{1}{m} \sum_{i=1}^m (ax^{(i)} + b - y^{(i)})$$

$$\begin{cases} a = a - \alpha \frac{\delta J(a, b)}{\delta a} \\ b = b - \alpha \frac{\delta J(a, b)}{\delta b} \end{cases}$$

$$\frac{\delta J(\theta)}{\delta \theta} = \frac{1}{m} X^T (X\theta - Y)$$

$$\theta = \theta - \alpha \frac{\delta J(\theta)}{\delta \theta}$$

$$X = \begin{bmatrix} x^{(1)} & 1 \\ x^{(2)} & 1 \\ \vdots & \vdots \\ x^{(m)} & 1 \end{bmatrix} \Rightarrow X^T = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

Notation Matriciel

EXEMPLE | Régression polynomiale

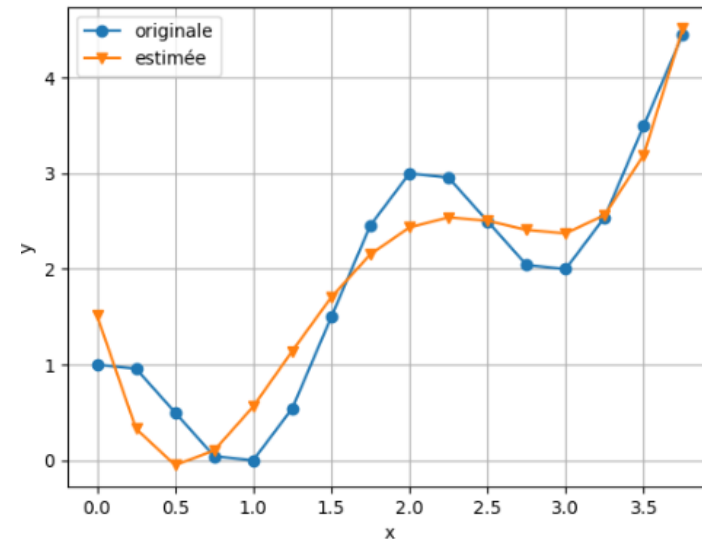
$$\Rightarrow f(x) = ax^2 + bx + c$$

En utilisant la notation matriciel le problème devient comme suite :

⇒ Le Modèle : $F = X\theta$

$$X = \begin{bmatrix} x^{(1)2} & x^{(1)} & 1 \\ x^{(2)2} & x^{(2)} & 1 \\ \vdots & \vdots & \vdots \\ x^{(m)2} & x^{(m)} & 1 \end{bmatrix} \text{ et } \theta = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

⇒ La fonction coût : $J(\theta) = \frac{1}{2m} \sum (X\theta - Y)^2$



MÉTHODE Gradient Descent

⇒ Le dérivé de la fonction coût :

$$\frac{\delta J(\theta)}{\delta \theta} = \frac{1}{m} X^T (X\theta - Y)$$

⇒ La mise à jour des paramètres :

$$\theta = \theta - \alpha \frac{\delta J(\theta)}{\delta \theta}$$

MÉTHODE DES MOINDRES CARRÉS

$$\theta = (X^T X)^{-1} X^T Y$$

Régression Linéaire Multiple

INTRODUCTION

- ↪ Extension naturelle de la régression linéaire simple.
- ↪ Utilisée quand plusieurs variables explicatives influencent une variable cible (y).

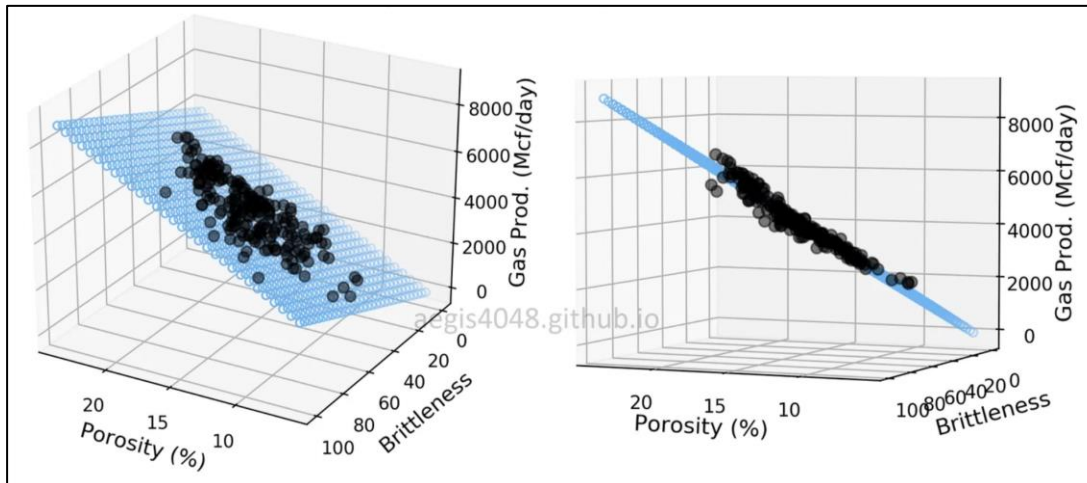
Exemple :

prédire le salaire en fonction de l'expérience, du niveau d'éducation et de l'âge.

Formule :

$$y = \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \beta_0$$

VISUALISATION 3D DE LA RÉGRESSION MULTIPLE



De la régression simple à la régression multiple

1

Variable explicative en régression simple

n

Variables explicatives en régression multiple

70%

Des modèles prédictifs utilisent la régression multiple

La régression linéaire multiple étend le modèle simple en permettant d'analyser l'influence de plusieurs variables explicatives sur une variable cible.

Cette approche est fondamentale dans l'analyse prédictive moderne et constitue la base de nombreuses techniques avancées d'apprentissage automatique.

Régression Linéaire Multiple

DATASET

Le dataset contient plusieurs variables explicatives

Variable cible (y)
Exemple :

- Variables explicatives :
Secteur, Niveau d'étude, années d'Expérience, Âge
- Variable cible (y) : Salaire (KDH)

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} & 1 \\ x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 \\ x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} & 1 \end{bmatrix} \quad Y = \begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ \vdots \\ y_m^{(1)} \end{bmatrix}$$

EN PRATIQUE

```
import pandas as pd
df = pd.read_csv("dataset_salaire_maroc.csv")
df.head()
```

Secteur	Niveau d'étude	Experience (années)	Âge	Salaire (KDH)
Privé	Bac	5	24	5.69
Multinationale	Bac+5	14	37	30.14
Multinationale	Sans	5	26	7.52
Privé	Sans	9	28	8.97
Privé	Bac	10	28	12.62

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Secteur'] = le.fit_transform(df['Secteur'])
df["Niveau d'étude"] = le.fit_transform(df["Niveau d'étude"])

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = df.drop(columns=["Salaire (KDH)"])
X_scaled = scaler.fit_transform(X)
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)
Y = df["Salaire (KDH)"]

X = np.hstack((X_scaled, np.ones((X_scaled.shape[0], 1))))
```

Régression Linéaire Multiple

MODÈLE

↳ Modèle général :

$$y = \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \beta_0$$

Chaque β_i mesure l'influence de x_i sur y .

↳ Exemple :

Salaire = 15 + 1.8(Secteur) + 2.5(Expérience) + 4.2(NiveauÉtude) + 0.8(Âge)

REPRÉSENTATION VECTORIELLE

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} & 1 \\ x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 \\ x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} & 1 \end{bmatrix} \quad \theta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \\ \beta_0 \end{bmatrix}$$

↳ Sous forme matricielle :

$$F = X\theta$$

X = matrice des observations

θ = vecteur des coefficients

F = vecteur des valeurs prédites

EN PRATIQUE

↳ Initialisation du vecteur θ avec des valeurs aléatoires

```
theta = np.random.randn(X.shape[1], 1)
```

↳ Faire des prédictions

```
def model(X, theta):
    | return X.dot(theta)

predictions = model(X, theta)
```

Régression Linéaire Multiple

FONCTION COÛT

↪ Erreur Quadratique Moyenne : Mesure de la qualité du modèle

$$J(\theta) = \frac{1}{2m} \sum (X\theta - Y)^2 = \frac{1}{2m} (X\theta - Y)^T (X\theta - Y)$$

EN PRATIQUE

↪ Version 1 : $\frac{1}{2m} \sum (X\theta - Y)^2$

```
def fonction_cout_v1(X, Y, theta):
    m = len(Y)
    predictions = model(X, theta)
    erreur = predictions - Y.values.reshape(-1, 1)
    cout = (1/(2*m)) * np.sum(erreur**2)
    return cout
fonction_cout_v1(X, Y, theta)
```

↪ Version 2 : $\frac{1}{2m} (X\theta - Y)^T (X\theta - Y)$

```
def fonction_cout_v2(X, y, theta):
    m = len(y)
    predictions = model(X, theta)
    part1 = (predictions - y.values.reshape(-1, 1)).T
    part2 = predictions - y.values.reshape(-1, 1)
    cout = (1/(2*m)) * part1.dot(part2)[0][0]
    return cout
fonction_cout_v2(X, Y, theta)
```

Il faut faire attention aux dimensions des matrices et/ou vecteurs :

↪ **predictions = model(X, theta)** → nous renvoie une matrice de dimension (m, 1) "c'est pour cela on change les dimensions du vecteur Y afin d'effectuer la soustraction"

↪ **Y** → dimension est (m,) → **Y.values.reshape(-1, 1)** → rendre les dimension (m,1) "(-1,1) pour forcer le format colonne et .values pour avoir un array numpy et pas une Series pandas »

↪ **.dot** → pour effectuer le produit matriciel



Régression Linéaire Multiple

ALGORITHME DE MINIMISATION - MOINDRES CARRÉS

↪ Résolution analytique :

$$\theta = (X^T X)^{-1} X^T Y$$

Solution exacte minimisant la fonction coût.

↪ Version 1: $\theta = (X^T X)^{-1} X^T Y$

```
theta_op = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(Y.values.reshape(-1, 1))
```



LinAlgError: Singular matrix



```
array([[ -2.19796999],
       [  0.45878237],
       [  4.7927427 ],
       [ -0.11970121],
       [13.66074   ]])
```

ALGORITHME DE MINIMISATION - DESCENTE DE GRADIENT

↪ Le dérivé de la fonction coût : $\frac{\delta J(\theta)}{\delta \theta} = \frac{1}{m} X^T (X\theta - Y)$

↪ La mise à jour des paramètres : $\theta = \theta - \alpha \frac{\delta J(\theta)}{\delta \theta}$

```
array([[ -2.20140058],
       [  0.48125645],
       [  3.3365764 ],
       [  1.33353596],
       [13.6602428 ]])
```



↪ $\frac{\delta J(\theta)}{\delta \theta} = \frac{1}{m} X^T (X\theta - Y)$

```
def calculer_derive(X, Y, theta):
    m = len(Y)
    predictions = model(X, theta)
    erreur = predictions - Y.values.reshape(-1, 1)
    derive = (1/m) * (X.T.dot(erreur))
    return derive
```

↪ $\theta = \theta - \alpha \frac{\delta J(\theta)}{\delta \theta}$

```
def gradient_descent(X, Y, theta, learning_rate=0.01, n_iterations=1000):
    for i in range(n_iterations):
        derive = calculer_derive(X, Y, theta)
        theta = theta - learning_rate * derive
    return theta
```



Évaluation de la qualité des modèles de régression

POURQUOI ÉVALUER UN MODÈLE?

↳ Objectif principal



Prédire avec précision de nouvelles données



Comprendre les relations entre variables

↳ Questions Clés



Mon modèle est-il fiable?



Comprendre les relations entre variables



Généralise-t-il bien sur de nouvelles données ?

LES 3 TYPES DE RÉGRESSION

↳ Simple

$$y = ax + b$$

- ▶ 1 variable explicative
- ▶ Relation linéaire

↳ Polynomiale

$$y = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

- ▶ 1 variable mais relation curviligne

↳ Multiple

$$y = a_1x_1 + a_2x_2 + \dots + a_nx_n + b$$

- ▶ Plusieurs variables explicatives
- ▶ Relations complexes

Régression Linéaire

EN UTILISANT LA BIBLIOTHÈQUE SKIT-LEARN



Importation

```
from sklearn.linear_model import LinearRegression
```

Création du modèle

```
model = LinearRegression()
```

Entrainement

```
model.fit(X, Y)
```

Faire des prédictions

```
predictions = model.predict(X)
```