



Introduction au Machine Learning

Séance 5 — Prédire un nombre : la régression linéaire, de l'équation normale à Ridge

Dr. El Hadji Bassirou TOURÉ

Département de Mathématiques et Informatique, Faculté des Sciences et
Techniques, Université Cheikh Anta Diop de Dakar

<https://ebtoure.sn/cours/introduction-machine-learning/chapitres/05-regression-lineaire.html>

Version du 11 octobre 2026



Version en ligne

Introduction au ML — Séance 5

Prédire un nombre : la régression linéaire, de l'équation normale à Ridge

Dr. El Hadji Bassirou TOURÉ

Département de Mathématiques et Informatique, Faculté des Sciences et Techniques, Université Cheikh Anta Diop de Dakar

Objectifs de la séance

Contenu

Cette séance construit le deuxième pilier du cours : prédire une **quantité** continue. Elle introduit le modèle linéaire, deux façons de l'entraîner — l'**équation normale** et la **descente de gradient** — puis les outils pour l'évaluer et le discipliner.

- Formuler la **régression** : la fonction f , la droite $\hat{y} = b + w x$, les résidus.
- Définir la **meilleure droite** par la **MSE**, et la calculer exactement : l'**équation normale**.
- Apprendre par itérations : la **descente de gradient**, pas calculés à la main, rôle du pas η .
- Lire **MAE**, **RMSE** et **R²** ; diagnostiquer par les résidus.
- Le degré d'un polynôme comme bouton de complexité ; la **courbe en U** ; **Ridge** pour dompter le sur-ajustement.

Prédire une quantité : le modèle linéaire

La régression : prédire un nombre

Changement de question

Les Séances 3 et 4 prédisaient une *classe* (séjour long ou court, oui ou non). La direction de l'hôpital de Thiès pose une autre question : **combien de jours** ce patient restera-t-il ? Une classe ne suffit plus — il faut un **nombre**.

$$f : \mathcal{X} \longrightarrow \mathbb{R}, \quad f(x_i) \approx y_i$$

La **régression** est l'apprentissage supervisé dont la sortie est un réel : à partir d'exemples étiquetés (x_i, y_i) avec $y_i \in \mathbb{R}$, f ne choisit plus une étiquette, elle renvoie une valeur sur un axe continu. Ici $\mathcal{X} \subset \mathbb{R}^5$ (âge, glycémie, hémoglobine, fièvre, saison) et la cible y est la **durée d'hospitalisation** en jours. Même protocole qu'en Séance 3 — **fit/predict**, train/test — seul l'espace de sortie change : $\mathcal{Y} = \mathbb{R}$ au lieu de $\{0, 1\}$, et l'erreur se mesure en *distance* $(\hat{y} - y)$, non plus en correct/incorrect.

Les données : X et un y devenu réel

Premières lignes réelles de DataSANTÉ-221

	âge	glycémie	hémoglobine	fièvre	saison	durée (j)
x_1	47	9,3	11,8	38,0	0	$y_1 = 3,8$
x_2	18	5,1	13,0	39,5	1	$y_2 = 0,5$
x_3	56	7,2	9,5	37,1	1	$y_3 = 6,5$

La matrice de design X est inchangée depuis la Séance 3 ; la seule nouveauté est dans y , qui vit désormais dans $\mathbb{R}^n$ et non dans $\{0, 1\}^n$.

Ce qui change depuis la Séance 3 : y vit dans $\mathbb{R}$

X et y de DataSANTÉ-221

$$X = \begin{pmatrix} 47 & 9,3 & 11,8 & 38,0 & 0 \\ 18 & 5,1 & 13,0 & 39,5 & 1 \\ 56 & 7,2 & 9,5 & 37,1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \in \mathbb{R}^{10000 \times 5} \quad y = \begin{pmatrix} 3,8 \\ 0,5 \\ 6,5 \\ \vdots \end{pmatrix} \in \mathbb{R}^{10000}$$

Exercice — régression ou classification ?

Énoncé

Pour chaque problème, déterminer s'il relève de la régression ou de la classification : **(a)** prédire le rendement d'un champ de mil à Kaolack (en tonnes/ha) ; **(b)** prédire si une transaction mobile money est frauduleuse ; **(c)** prédire la glycémie d'un patient dans six mois ; **(d)** prédire la gravité d'un cas parmi {bénin, modéré, sévère}.

Correction

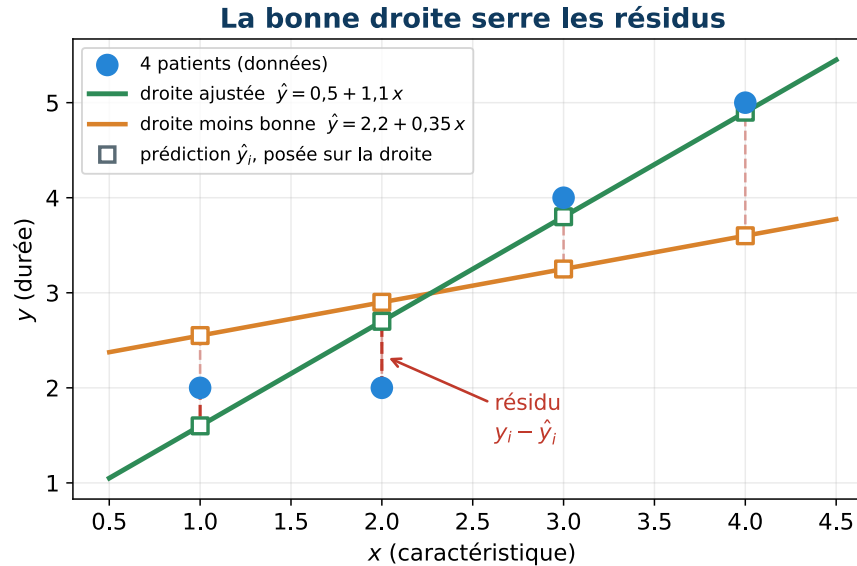
(a) régression — la sortie est un réel ; **(b)** classification binaire ; **(c)** régression — un nombre, même si l'entrée contient l'ancienne glycémie ; **(d)** classification à trois classes : les modalités sont *ordonnées* mais restent des catégories, pas des quantités continues.

L'idée : une droite qui suit le nuage

Analogie

Le prix d'une course de taxi à Dakar : une *prise en charge* fixe, puis un tarif *par kilomètre*. Le modèle linéaire fait de chaque prédiction une course de taxi : un point de départ b , et un prix w_j par unité de chaque caractéristique.

Deux droites, et le résidu qui les départage



Même nuage, deux droites de qualités différentes : le carré $\hat{y}_i$ est posé *sur* la droite, le tiret rouge est le **résidu** $y_i - \hat{y}_i$; la bonne droite serre les résidus.

Le modèle linéaire, formellement

Définition

Le **modèle linéaire** prédit par une somme pondérée des caractéristiques : chaque variable contribue proportionnellement à son poids, et b fixe le niveau de base.

$$\hat{y} = b + w_1x_1 + w_2x_2 + \cdots + w_dx_d = b + \mathbf{w} \cdot \mathbf{x}$$

$\mathbf{w} = (w_1, \dots, w_d)$ est le vecteur des **poids** (un par caractéristique) et b l'**ordonnée à l'origine** (*intercept*) : la prédiction quand toutes les caractéristiques valent 0. Apprendre, c'est choisir les $d + 1$ nombres $(b, w_1, \dots, w_d)$ — rien d'autre. Avec une seule variable, le modèle est la droite $\hat{y} = b + w x$.

Exemple — évaluer $\hat{y}$ terme à terme

Le patient x_1 passe dans le modèle

Modèle entraîné sur DataSANTÉ-221 (coefficients réels, arrondis) : $b = 0,64$, puis par variable :

	âge	glycémie	hémoglobine	fièvre	saison
poids w_j	0,048	0,566	-0,263	0,011	1,379
valeur x_{1j}	47	9,3	11,8	38,0	0
produit $w_j x_{1j}$	2,27	5,26	-3,11	0,41	0

$\hat{y}_1 = 0,64 + 2,27 + 5,26 - 3,11 + 0,41 + 0 = 5,48$ jours. La durée réelle est $y_1 = 3,8$:
résidu = $3,8 - 5,48 = -1,68$ jour — le modèle surestime ce patient.

Lire les coefficients : chaque poids est une pente

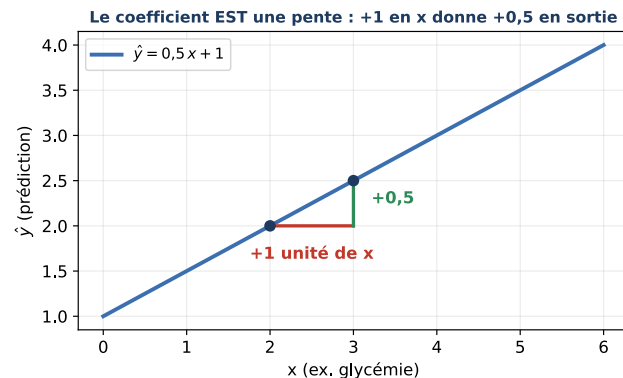
L'idée en une phrase

Chaque poids w_j est un **taux de changement** : « une unité de plus sur la variable j déplace la prédiction de w_j » (toutes choses égales par ailleurs).

Sur la durée d'hospitalisation :

- glycémie $w = +0,57$: +1 mmol/L ajoute $\approx$ une demi-journée ;
- saison $w = +1,38$: l'hivernage ajoute 1,4 jour ;
- hémoglobine $w = -0,26$: meilleure hémoglobine, séjour plus court.

Piège. Comparer la *taille* de deux coefficients n'a de sens que sur la **même échelle** (Séance 4) ; et un poids décrit une *association*, pas une cause.



Exercice — manier les coefficients

Énoncé

Avec le modèle ci-dessus ($w_{\text{glycémie}} = 0,566$, $w_{\text{saison}} = 1,379$) : **(a)** de combien la durée prédite change-t-elle si la glycémie augmente de 2 mmol/L, le reste étant inchangé ? **(b)** et si, de plus, on passe de la saison sèche (0) à l'hivernage (1) ?

Correction

(a) $\Delta\hat{y} = 0,566 \times 2 = +1,13$ jour. **(b)** s'y ajoute $1,379 \times 1$, soit au total $1,13 + 1,38 = +2,51$ jours. La linéarité rend ces effets *additifs* : chaque variable pousse la prédiction indépendamment des autres.

La meilleure droite : la MSE

Que veut dire la meilleure droite ?

L'idée en une phrase

Une droite est bonne si ses erreurs verticales — les **résidus** $y_i - \hat{y}_i$ — sont petites ; la meilleure droite est celle qui rend ces erreurs *globalement* les plus petites possibles.

Il faut donc une règle qui transforme n résidus en **un seul nombre** à minimiser. La convention du ML : élever chaque résidu au carré (les signes ne se compensent plus, les grandes erreurs pèsent lourd), puis moyenner.

L'erreur quadratique moyenne, formellement

Définition

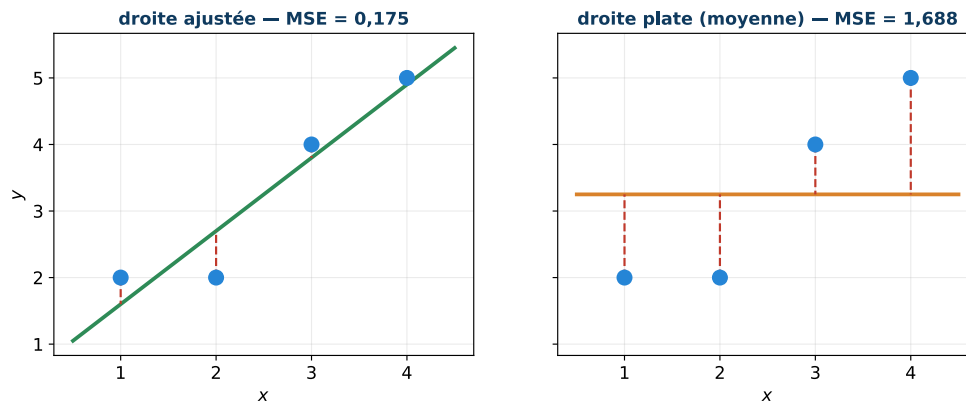
La **MSE** (*mean squared error*) d'un modèle est la moyenne des carrés de ses résidus. Entraîner une régression linéaire, c'est chercher les paramètres qui la minimisent : le critère des **moindres carrés**.

$$J(b, \mathbf{w}) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (\hat{b}, \hat{\mathbf{w}}) = \arg \min_{b, \mathbf{w}} J(b, \mathbf{w})$$

Chaque terme $(y_i - \hat{y}_i)^2$ est le carré d'un résidu ; la somme agrège, la division par n moyenne. J dépend des *paramètres* (les données sont fixées) : changer $(b, \mathbf{w})$ déplace la droite et change J . Le carré rend J **lisse** et en forme de **cuvette** — propriété décisive pour la suite.

Exemple — comparer deux droites par leur MSE

Deux droites candidates : la MSE départage



Jouet : $x = [1, 2, 3, 4]$, $y = [2, 2, 4, 5]$. À gauche, $\hat{y} = 0,5 + 1,1 x$: résidus (0,4; -0,7; 0,2; 0,1), carrés (0,16; 0,49; 0,04; 0,01), $MSE = 0,70/4 = 0,175$. À droite, la droite plate $\hat{y} = \bar{y} = 3,25$: $MSE = 1,688$ — dix fois pire.

Exercice — calculer la MSE de la droite plate

Énoncé

Vérifier la valeur 1,688 : calculer les quatre résidus de la droite $\hat{y} = 3,25$ sur le jouet $y = [2, 2, 4, 5]$, leurs carrés, puis la MSE.

Correction

Résidus : $2 - 3,25 = -1,25$; $-1,25$; $0,75$; $1,75$. Carrés : $1,5625$; $1,5625$; $0,5625$; $3,0625$.
Somme = $6,75$, $MSE = 6,75/4 = 1,6875$. À noter : ce $6,75$ resservira tel quel dans le calcul du R^2 .

L'équation normale : le minimum exact

Le cas d'une variable, formellement

L'idée en une phrase

La MSE est une cuvette parfaitement régulière : au fond, la pente est nulle dans toutes les directions — et ce point d'équilibre se calcule **directement** à partir des données, sans tâtonner. À une variable, il se lit dans les écarts aux moyennes $\bar{x}$ et $\bar{y}$.

$$\hat{w} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}, \quad \hat{b} = \bar{y} - \hat{w} \bar{x}$$

Le numérateur de $\hat{w}$ mesure comment x et y *varient ensemble* ; le dénominateur, combien x varie tout seul. Une fois la pente connue, $\hat{b}$ cale la droite pour qu'elle passe exactement par le centre de gravité $(\bar{x}, \bar{y})$ du nuage.

Exemple — la droite calculée à la main

Sur les quatre patients du jouet

$x = [1, 2, 3, 4], y = [2, 2, 4, 5]$; moyennes $\bar{x} = 2,5, \bar{y} = 3,25$.

i	$x_i - \bar{x}$	$y_i - \bar{y}$	produit	$(x_i - \bar{x})^2$
1	-1,5	-1,25	+1,875	2,25
2	-0,5	-1,25	+0,625	0,25
3	+0,5	+0,75	+0,375	0,25
4	+1,5	+1,75	+2,625	2,25
Σ			5,5	5,0

$\hat{w} = 5,5/5,0 = 1,1, \hat{b} = 3,25 - 1,1 \times 2,5 = 0,5$ — la droite verte des diapositives précédentes. `scikit-learn` confirme : `coef_` = 1,1, `intercept_` = 0,5.

L'équation normale, formellement

Définition

En dimension quelconque, on absorbe b dans les poids en ajoutant à X une **colonne de 1** ; le minimum de la MSE est alors donné par une formule matricielle unique : l'**équation normale**.

$$\tilde{X} = \begin{pmatrix} 1 & x_1^\top \\ \vdots & \vdots \\ 1 & x_n^\top \end{pmatrix}, \quad \hat{\theta} = (\tilde{X}^\top \tilde{X})^{-1} \tilde{X}^\top y, \quad \hat{\theta} = (\hat{b}, \hat{w}_1, \dots, \hat{w}_d)$$

Lecture terme à terme : $\tilde{X}^\top y$ confronte chaque colonne aux cibles (qui co-varie avec y) ; $\tilde{X}^\top \tilde{X}$ enregistre comment les colonnes varient entre elles ; l'inverse $(\cdot)^{-1}$ démêle leurs contributions. Cette formule est le point où toutes les pentes de J s'annulent — on l'admet, et on va la **vérifier** sur le jouet.

Exemple — l'équation normale sur le jouet, étape 1

Construire les deux blocs

Jouet à une variable : la matrice augmentée $\tilde{X}$ a deux colonnes (les 1, puis x).

$$\tilde{X} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix}, \quad \tilde{X}^\top \tilde{X} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}, \quad \tilde{X}^\top y = \begin{pmatrix} 13 \\ 38 \end{pmatrix}$$

Chaque case se lit : $4 = \text{nombre de patients}$, $10 = \sum x_i$, $30 = \sum x_i^2$; côté cibles, $13 = \sum y_i$ et $38 = \sum x_i y_i$. Tout est déjà là : effectifs, sommes, co-variations.

Exemple — l'équation normale sur le jouet, étape 2

Inverser puis multiplier

Inverse d'une matrice 2×2 : échanger la diagonale, changer le signe du reste, diviser par le déterminant $4 \times 30 - 10 \times 10 = \mathbf{20}$:

$$(\tilde{X}^\top \tilde{X})^{-1} = \frac{1}{20} \begin{pmatrix} 30 & -10 \\ -10 & 4 \end{pmatrix} = \begin{pmatrix} 1,5 & -0,5 \\ -0,5 & 0,2 \end{pmatrix}$$

$$\hat{\theta} = \begin{pmatrix} 1,5 & -0,5 \\ -0,5 & 0,2 \end{pmatrix} \begin{pmatrix} 13 \\ 38 \end{pmatrix} = \begin{pmatrix} 1,5 \times 13 - 0,5 \times 38 \\ -0,5 \times 13 + 0,2 \times 38 \end{pmatrix} = \begin{pmatrix} 0,5 \\ 1,1 \end{pmatrix}$$

Le même $(\hat{b}, \hat{w}) = (0,5; 1,1)$ que les formules fermées — l'équation normale les généralise à d variables. [LinearRegression](#) applique exactement ce calcul.

Exercice — résoudre une équation normale

Énoncé

Sur un autre jeu, on a calculé $\tilde{X}^\top \tilde{X} = \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix}$ et $\tilde{X}^\top y = \begin{pmatrix} 4 \\ 7 \end{pmatrix}$. Calculer $\hat{\theta} = (\hat{b}, \hat{w})$.

Correction

Déterminant $= 2 \times 5 - 3 \times 3 = 1$, donc l'inverse est $\begin{pmatrix} 5 & -3 \\ -3 & 2 \end{pmatrix}$. Produit : $\hat{b} = 5 \times 4 - 3 \times 7 = -1$ et $\hat{w} = -3 \times 4 + 2 \times 7 = 2$. La droite des moindres carrés est $\hat{y} = -1 + 2x$.

Piège — quand l'inversion se dérobe

Deux limites de la solution exacte

Colinéarité : si une colonne de X est (presque) une combinaison des autres — le poids en kg *et* en livres — alors $\tilde{X}^\top \tilde{X}$ n'est pas inversible : une infinité de droites donnent la même MSE, le calcul devient instable. **Coût** : inverser une matrice $(d + 1) \times (d + 1)$ devient lourd quand d se compte en milliers.

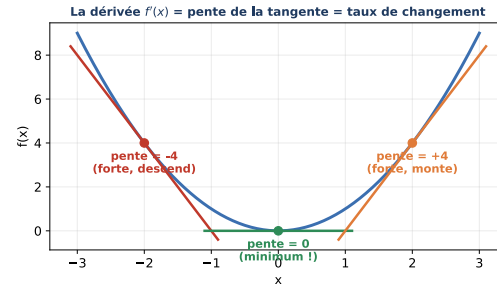
Dans les deux cas, la parade est la même : renoncer au calcul exact et **approcher** le minimum par petits pas. C'est la descente de gradient — la méthode d'apprentissage la plus importante de tout le ML, des droites aux réseaux de neurones.

La descente de gradient

La dérivée : mesurer une pente

Pourquoi, et l'idée en une phrase

L'équation normale exige d'**inverser une matrice** — impraticable en grande dimension, impossible pour les réseaux. On cherche alors le minimum à tâtons (la **descente de gradient**), et pour « descendre » on mesure une pente : la **dérivée** $f'(x)$, taux de changement de f . Positive, f monte ; négative, f descend ; **nulle**, on est au creux.

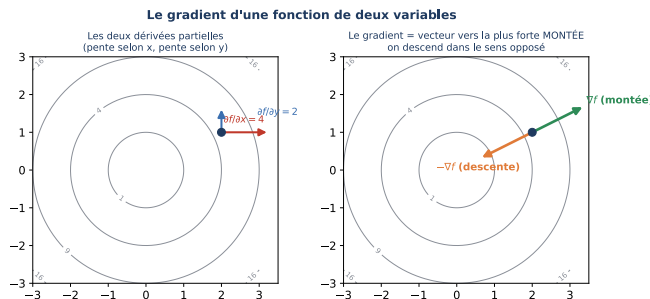


Pente -4 (descend), $+4$ (monte), 0 au creux : le *minimum* est là où la dérivée s'annule.

Dérivées partielles et gradient

Une pente par variable, un vecteur pour les réunir

Quand f dépend de plusieurs variables, la **dérivée partielle** $\partial f / \partial x$ mesure le taux de changement selon *une seule* composante, les autres figées (sur une colline : la pente vers l'est, pas vers le nord). Le **gradient** $\nabla f = (\partial f / \partial x, \partial f / \partial y)$ rassemble toutes ces pentes en un **vecteur** qui pointe vers la *plus forte montée* ; sa longueur dit à quel point ça monte.

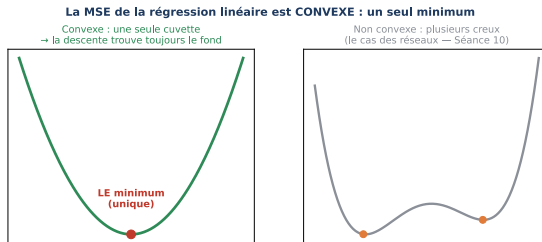


$\nabla f = (4; 2)$ (vert) pointe vers la plus forte montée ; on descend dans le sens opposé $-\nabla f$ (orange).

La perte est une cuvette : descendre pas à pas

Pourquoi la descente marche ici

La MSE de la régression linéaire est **convexe** : sa surface est une *cuvette* à **un seul** creux. D'où qu'on vienne, suivre la pente vers le bas mène *toujours* au même fond — le minimum global. C'est la stratégie d'une randonneuse dans le brouillard : sentir la pente, faire un pas vers le bas, recommencer ; la pente locale suffit.



Convexe (gauche) : un seul creux. Non convexe (droite) : plusieurs creux (le cas des réseaux, Séance 10).

La règle de mise à jour, formellement

Définition

On applique au critère J le gradient qu'on vient de définir : ∇J pointe vers la plus forte montée de l'erreur ; la règle de mise à jour avance dans le sens *opposé*, d'un pas de taille η .

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla J(\theta^{(t)}), \quad \theta = (b, \mathbf{w})$$

Le signe « $-$ » est tout l'algorithme : le gradient montre la montée, on prend son opposé pour descendre. Le **pas d'apprentissage** η (*learning rate*) dose la prudence : la randonneuse fait-elle des pas de fourmi ou des bonds ? On répète jusqu'à ce que le gradient devienne quasi nul — le fond de la cuvette.

Le gradient de la MSE, formellement

Les deux pentes de la cuvette

Pour la droite $\hat{y} = b + w x$, le gradient de la MSE a une forme remarquablement lisible : ce sont des **moyennes de résidus**.

$$\frac{\partial J}{\partial b} = -\frac{2}{n} \sum_{i=1}^n r_i, \quad \frac{\partial J}{\partial w} = -\frac{2}{n} \sum_{i=1}^n x_i r_i, \quad r_i = y_i - \hat{y}_i$$

(Rappel juste-à-temps : dériver $(y_i - b - wx_i)^2$ par b donne $-2(y_i - b - wx_i)$ — la règle de la fonction composée, rien de plus.) Lecture : si les résidus sont en moyenne positifs, le modèle prédit trop bas $\Rightarrow$ la pente sur b est négative $\Rightarrow$ la mise à jour *augmente* b . Le gradient transforme les erreurs en correction.

Exemple — l'itération 1, à la main

Départ au point zéro

Jouet, départ $(b, w) = (0, 0)$, pas $\eta = 0,1$. Le modèle prédit $\hat{y}_i = 0$ partout, donc $r = y = (2, 2, 4, 5)$.

$$\frac{\partial J}{\partial b} = -\frac{2}{4}(2 + 2 + 4 + 5) = -\frac{2}{4} \times 13 = -6,5$$

$$\frac{\partial J}{\partial w} = -\frac{2}{4}(1 \cdot 2 + 2 \cdot 2 + 3 \cdot 4 + 4 \cdot 5) = -\frac{2}{4} \times 38 = -19$$

Mise à jour : $b = 0 - 0,1 \times (-6,5) = 0,65$; $w = 0 - 0,1 \times (-19) = 1,9$. La MSE chute de 12,25 à 5,60 : un seul pas, déjà moitié moins d'erreur.

Exemple — itérations 2 et 3 : le zigzag

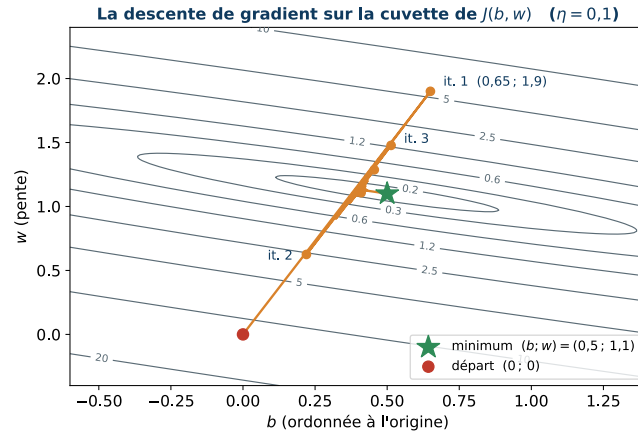
La correction se corrige

Au point (0,65 ; 1,9), la droite est devenue *trop pentue* : les résidus changent de signe, le gradient s'inverse.

it.	$\partial J / \partial b$	$\partial J / \partial w$	b	w	J
1	-6,5	-19	0,65	1,9	5,60
2	+4,3	+12,75	0,22	0,625	2,61
3	-2,94	-8,53	0,51	1,48	1,27

Le trajet **zigzague** de part et d'autre du minimum — sur-correction, contre-correction — mais chaque rebond est plus petit que le précédent, et J ne cesse de baisser.

La trajectoire et la convergence



Du zigzag au minimum exact

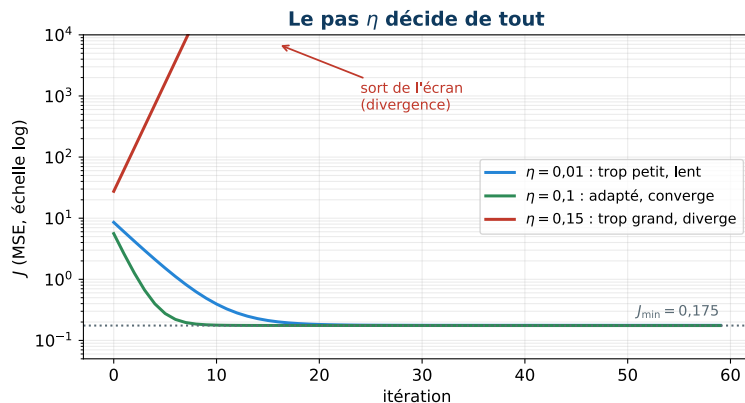
Itérations, $\eta = 0,1$:

it.	b	w	J
10	0,410	1,107	0,180
100	0,495	1,102	0,175
300	0,500	1,100	0,175

À l'itération 300, la descente retrouve $(0,5; 1,1)$ et $J = 0,175$: **exactement** la solution de l'équation normale.

Courbes de niveau de $J(b, w)$: départ rouge en $(0; 0)$, zigzag orange, convergence vers l'étoile $(0,5; 1,1)$. Deux chemins, un même fond — la cuvette garantit qu'aucun autre creux ne piège la descente.

Le pas η décide de tout



Trois réglages, trois destins : $\eta = 0,01$ converge mais lambine encore loin du but après 60 itérations ; $\eta = 0,1$ atteint le fond en une vingtaine de pas ; $\eta = 0,15$ **diverge** — chaque bond dépasse tellement le minimum que J atteint $1,5 \times 10^8$ dès l'itération 20.

Exercice — calculer une itération

Énoncé

Reprendre le départ $(b, w) = (0, 0)$ sur le jouet, mais avec un pas $\eta = 0,05$. Calculer (b, w) après l'itération 1. (Les gradients au point de départ sont ceux de l'exemple : $-6,5$ et -19 .)

Correction

Les gradients ne dépendent que du point courant, pas de η : $b = 0 - 0,05 \times (-6,5) = \mathbf{0,325}$ et $w = 0 - 0,05 \times (-19) = \mathbf{0,95}$. Même direction que l'exemple, pas deux fois plus court : η ne choisit pas *où* aller, seulement *jusqu'où* avancer.

En pratique : deux solveurs pour un même minimum

```
from sklearn.linear_model import LinearRegression, SGDRegressor

exact = LinearRegression()          # equation normale (solution exacte)
exact.fit(X_train, y_train)

iteratif = SGDRegressor(eta0=0.01, max_iter=1000, random_state=0)
iteratif.fit(X_train_std, y_train)  # descente de gradient (par pas)
```

`LinearRegression` résout l'équation normale ; `SGDRegressor` descend le gradient (variante *stochastique* : le pas est estimé sur un sous-échantillon, même principe). Petite dimension $\Rightarrow$ solution exacte ; très grande dimension ou flux de données $\Rightarrow$ descente. Les réseaux de neurones n'auront *que* la descente.

Piège — descendre sans standardiser

La cuvette étirée

Si l'âge varie de 5 à 85 et la glycémie de 3 à 18, la cuvette de J est un **canyon étroit** : très pentue dans une direction, presque plate dans l'autre. Aucun η ne convient aux deux à la fois — la descente zigzague violemment ou rampe. La **standardisation** (Séance 4) rend la cuvette ronde : c'est une condition de bon fonctionnement de la descente, comme elle l'était pour le kNN.

D'où le `X_train_std` du code précédent : `SGDRegressor` se place toujours *après* un `StandardScaler` dans le Pipeline.

Mesurer : MAE, RMSE, R^2 — et le verdict sur DataSANTÉ-221

MAE et RMSE, formellement

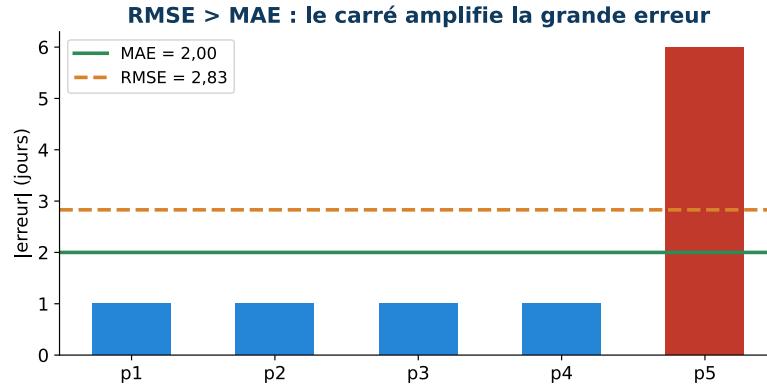
Deux moyennes d'erreur

L'accuracy n'existe plus — aucune prédiction continue n'est exactement juste. On moyenne la *taille* des erreurs : en valeur absolue (**MAE**), ou en passant par les carrés (**RMSE**).

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad \text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

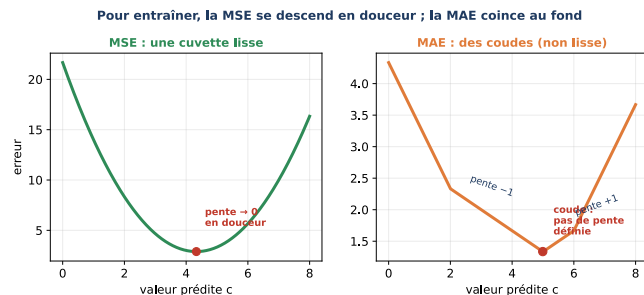
Les deux s'expriment dans l'**unité de la cible** (des jours) — la racine de la RMSE défait le carré de la MSE. La MAE traite toutes les erreurs au même tarif ; la RMSE, héritière du carré, **surtaxe les grandes erreurs**. Toujours $\text{RMSE} \geq \text{MAE}$, et l'écart entre les deux signale des erreurs très inégales.

Exemple — pourquoi la RMSE dépasse la MAE



Cinq patients, erreurs absolues (1, 1, 1, 1, 6). $MAE = 10/5 = \mathbf{2,00}$ jours. $RMSE = \sqrt{(1 + 1 + 1 + 1 + 36)/5} = \sqrt{8} = \mathbf{2,83}$ jours. La cinquième erreur pèse 36 dans les carrés contre 6 dans les valeurs absolues : un seul patient raté tire la RMSE bien au-dessus de la MAE.

Mesurer avec la MAE, optimiser avec la MSE



Deux rôles, deux choix

- **MAE** : toutes les erreurs au même tarif, robuste aux valeurs extrêmes et lisible en jours — mais sa courbe forme un **coude** au minimum (pente indéfinie), malcommode à minimiser.
- **MSE** : surtaxe les grosses erreurs (sensible aux extrêmes) — mais sa cuvette est **lisse et dérivable partout**, facile à descendre.
- En pratique : **entraîner** en minimisant la MSE, puis **rapporter** MAE et RMSE (lecture en jours).

La MSE (gauche) a une pente qui s'annule en douceur ; la MAE (droite) a un coude où la pente saute de -1 à $+1$ sans passer par 0.

Le R^2 , formellement

L'idée en une phrase

Le R^2 compare le modèle au prédicteur le plus paresseux — toujours répondre la moyenne $\bar{y}$ — et mesure quelle part de l'erreur de ce paresseux le modèle fait disparaître.

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Numérateur : l'erreur (quadratique) du modèle ; dénominateur : celle de la moyenne. $R^2 = 1$: prédiction parfaite ; $R^2 = 0$: pas mieux que la moyenne ; $R^2 < 0$: pire que la moyenne — toujours possible sur le test, et toujours mauvais signe. Sans unité, le R^2 complète la MAE/RMSE qui, elles, parlent en jours.

Exemple — le R^2 du jouet, décomposé

Les deux sommes sont déjà connues

Pour la droite $\hat{y} = 0,5 + 1,1x$: la somme des carrés des résidus vaut 0,70 (calcul de la MSE).
Pour la moyenne : 6,75 (l'exercice de la droite plate).

$$R^2 = 1 - \frac{0,70}{6,75} = 1 - 0,104 = 0,896$$

La droite efface 89,6 % de l'erreur quadratique du prédicteur-moyenne. `r2_score` de `scikit-learn` retourne la même valeur — et c'est ce que `score()` calcule pour tout régresseur.

Exercice — interpréter trois R^2

Énoncé

Trois modèles de la durée d'hospitalisation affichent, sur le *test* : $R_A^2 = 0,90$, $R_B^2 = 0,00$, $R_C^2 = -0,30$. Interpréter chaque valeur en une phrase.

Correction

A supprime 90 % de l'erreur quadratique de la moyenne : excellent sur ces données. *B* ne fait *pas mieux* que répondre $\bar{y}$ partout : il n'a rien appris d'utile. *C* fait *pire* que la moyenne — typique d'un modèle sur-ajusté à son train, ou d'une fuite dans le protocole. Réflexe : un R^2 s'annonce toujours *avec* la MAE ou la RMSE, qui parlent en jours.

La régression de la durée d'hospitalisation

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, r2_score

X = df[["age", "glycemie", "hemoglobine", "fievre", "saison"]]
y = df["duree_hospit_j"] # cible continue, en jours
X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.2,
                                          random_state=42)

modele = LinearRegression().fit(X_tr, y_tr)
y_pred = modele.predict(X_te)
```

Le protocole de la Séance 3, à deux différences près : la cible est `duree_hospit_j` (pas de `stratify` — il ne concerne que les classes), et les métriques changent de famille.

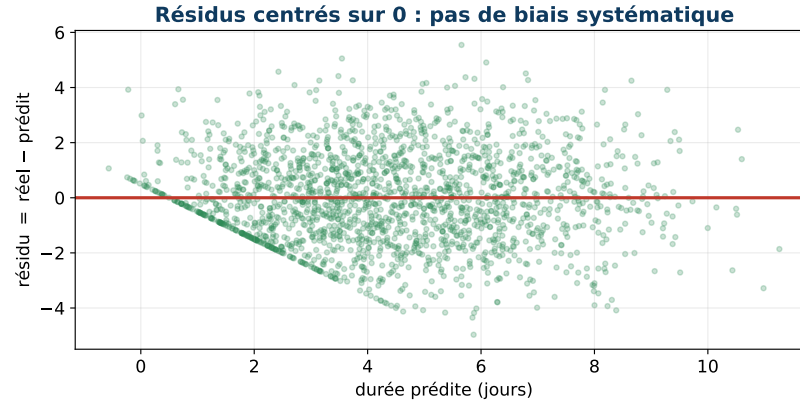
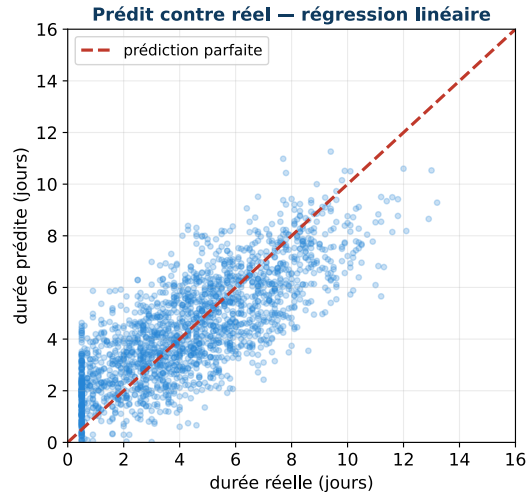
Le bilan chiffré de la séance

Trois modèles sur le test

modèle	MAE (jours)	RMSE (jours)	R ²
baseline (moyenne)	2,17	—	—0,001
arbre de régression, prof. 5	1,45	—	0,533
régression linéaire	1,35	1,67	0,598

Le linéaire se trompe de 1,35 jour en moyenne et efface 60 % de l'erreur de la baseline. Il **devance l'arbre** de la Séance 3 : la durée est fabriquée par des effets additifs (glycémie, saison...), terrain naturel d'un modèle linéaire — chaque famille de modèles a ses terrains.

Le diagnostic visuel : prédit contre réel, résidus



Ce que disent ces deux nuages

À gauche : le nuage longe la diagonale « prédiction parfaite » — avec un écrasement sur les longues durées, que le modèle sous-estime. À droite : les résidus en fonction du prédit, centrés sur 0 (pas de biais systématique). Toute *forme* dans ce nuage — courbure, entonnoir — signalerait un motif que la droite n'a pas capté.

Le degré : sous- et sur-ajustement

Au-delà de la droite : l'expansion polynomiale

Définition

Quand la relation est *courbe*, on garde le modèle linéaire mais on enrichit les **entrées** : aux côtés de x , on ajoute $x^2, x^3, \dots, x^p$ comme nouvelles colonnes. La droite, tracée dans cet espace enrichi, devient une courbe dans l'espace d'origine.

$$\varphi_p(x) = (x, x^2, \dots, x^p) \quad \hat{y} = b + w_1x + w_2x^2 + \dots + w_px^p$$

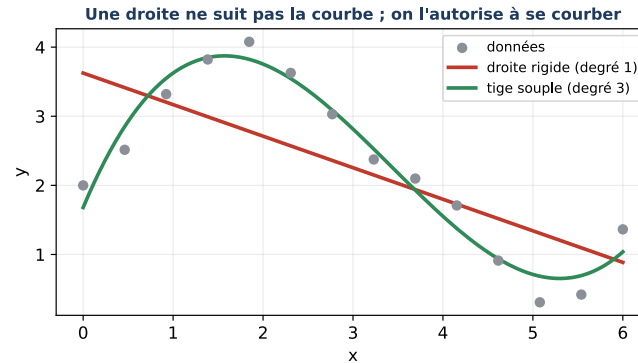
Exemple : $\varphi_3(2) = (2, 4, 8)$. Le modèle reste *linéaire en ses poids* — équation normale et descente de gradient s'appliquent inchangées. Le **degré** p devient un bouton de complexité, exactement comme la profondeur de l'arbre en Séance 3 — et **PolynomialFeatures** est une étape de Pipeline, comme tout prétraitement de la Séance 4.

L'analogie : une règle qu'on laisse se courber

Comprendre la régression polynomiale

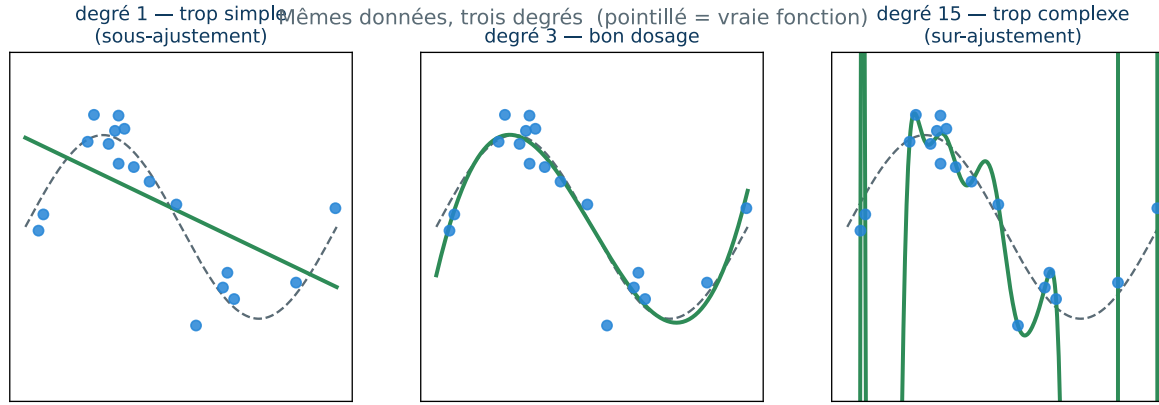
Une droite, c'est une règle **rigide** : elle ne peut pas épouser une courbe. Lui ajouter $x^2, x^3, \dots$ revient à remplacer la règle par une **tige souple** : plus on autorise de courbures (le degré p), plus elle suit les points. Trop souple, elle ondule pour passer par *chaque* point, jusqu'à suivre le bruit — c'est le sur-ajustement.

La règle rigide et la tige souple, en image



La droite rigide (rouge) rate la courbure des points ; le polynôme de degré 3 (vert), plus souple, la suit.

Trois degrés, trois destins



Trente points tirés d'une sinusoïde bruitée (pointillés : la vraie fonction). Le degré 1 est trop rigide pour la courbure : **sous-ajustement**. Le degré 3 épouse la forme. Le degré 15 passe presque par chaque point — il apprend le *bruit* : **sur-ajustement**. Le vocabulaire de la Séance 3 s'applique tel quel ; seul le bouton a changé.

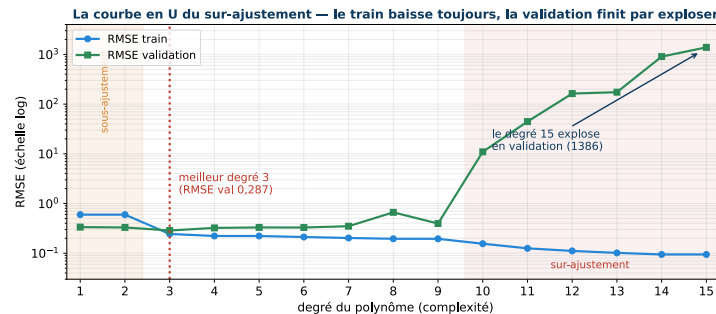
L'erreur d'entraînement est une flatteuse

Le piège chiffré

Sinusoïde (18 train, 12 validation) :

	RMSE train	RMSE val.
degré 3	0,244	0,287
degré 15	0,094	1386

Au train seul, le degré 15 semble « deux fois et demie meilleur » ; en validation il est **cinq mille fois pire**. Seule une donnée *jamais vue* dit la vérité.



Degrés 1 à 15 : la RMSE train (bleu) descend *toujours* ; la validation (vert) creuse au degré 3, puis explose. Choisir la complexité, c'est chercher le creux du U — jamais le bas de la courbe bleue.

Exercice — diagnostiquer par la paire de scores

Énoncé

Trois polynômes sur un même jeu : A — RMSE train 0,60, validation 0,62 ; B — train 0,24, validation 0,29 ; C — train 0,09, validation 14,5. Diagnostiquer chacun (sous-ajusté, bien dosé, sur-ajusté) et choisir le modèle à déployer.

Correction

A : deux erreurs *hautes et proches* — trop simple, sous-ajustement. C : train minuscule, validation catastrophique — mémorisation, sur-ajustement. B : erreurs basses et voisines — le bon dosage, c'est lui que l'on déploie. Le diagnostic se lit toujours dans l'*écart* entre les deux scores, jamais dans le train seul.

Ridge : dompter le sur-ajustement

L'idée : punir les coefficients géants

L'empreinte du sur-ajustement

Le polynôme de degré 15 qui explose en validation cache un symptôme spectaculaire : son plus grand coefficient atteint **9,2 milliards**. Pour zigzaguer entre 18 points, la courbe a besoin de pentes vertigineuses qui se compensent au micron près — c'est cela, mémoriser du bruit.

L'idée en une phrase

Ridge ajoute à l'erreur une **pénalité qui grandit avec la taille des coefficients** — comme un *élastique* reliant chaque coefficient à zéro : le poids ne glisse plus tout à fait jusqu'au minimum MSE, il s'en *approche* en restant plus près de 0. Le modèle ne garde alors un gros coefficient que s'il réduit *vraiment* l'erreur ; les superflus sont tirés vers zéro, et la course aux pentes géantes n'est plus rentable.

Ridge, formellement

Définition

Ridge minimise la MSE *augmentée* d'une pénalité proportionnelle à la somme des carrés des poids, dosée par $\alpha \geq 0$.

$$J_{\text{ridge}}(b, \mathbf{w}) = \underbrace{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}_{\text{coller aux données}} + \alpha \underbrace{\sum_{j=1}^d w_j^2}_{\text{rester sobre}}$$

Les deux termes tirent en sens contraires et α arbitre : $\alpha = 0$ redonne les moindres carrés ordinaires ; α immense écrase tous les poids vers 0 (la droite plate). b n'est pas pénalisé — le niveau de base n'est pas un luxe. La cuvette reste une cuvette : équation normale et descente de gradient s'adaptent sans douleur.

Comment la pénalité agit

Comment ça pénalise, concrètement

On ne minimise plus l'erreur seule, mais la somme $erreur + \alpha \sum_j w_j^2$. Augmenter un poids w_j a donc deux effets opposés : cela peut **baisser l'erreur** (gain), mais cela **fait toujours monter la pénalité** (coût, en w_j^2). À l'optimum, chaque poids s'arrête là où le gain marginal sur l'erreur *égale* le coût marginal de la pénalité. Un poids inutile n'apporte aucun gain : seul son coût compte, et il est poussé vers 0.

Le carré rend la pénalité *progressive* : doubler un coefficient quadruple son coût. Les très gros poids — ceux du sur-ajustement — deviennent les plus chers, donc les premiers sabrés. C'est α qui fixe le prix : plus il est grand, plus les coefficients sont comprimés.

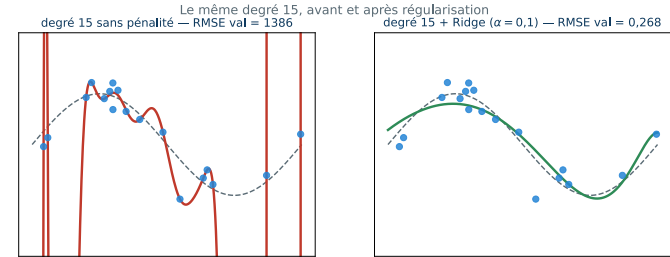
Le degré 15 dompté : avant et après

Le même polynôme sous quatre régimes

Degré 15 sur la sinusoïde, coefficients standardisés :

α	plus grand $ w_j $	RMSE val.
0	9,2 milliards	1386
0,001	7,8	0,31
0,1	1,8	0,27
100	0,05	0,51

À $\alpha = 0,1$, le degré 15 régularisé (0,27) bat même le meilleur degré non pénalisé (0,287).



Même degré 15 : sans pénalité la courbe s'emballe ; avec $\alpha = 0,1$ elle retrouve la sinusoïde.

Exercice — le rôle de α

Énoncé

(a) Vers quel modèle tend Ridge quand $\alpha \rightarrow \infty$? **(b)** Un collègue règle α en choisissant la valeur qui minimise la RMSE *d'entraînement*. Quelle valeur va-t-il systématiquement choisir, et pourquoi est-ce une erreur ?

Correction

(a) Tous les w_j sont écrasés vers 0 : il reste $\hat{y} = b$, la droite plate à hauteur de la moyenne — le sous-ajustement maximal. **(b)** $\alpha = 0$: toute pénalité ne peut qu'augmenter l'erreur de train, puisqu'elle contraint l'ajustement. Le bon α se choisit sur la **validation** — le train, flatteur, vote toujours pour la complexité maximale.

Piège — régulariser sans standardiser

Une pénalité injuste

La pénalité $\alpha \sum w_j^2$ compare les poids entre eux — elle n'a de sens que si les variables parlent la **même monnaie**. Sans standardisation, une variable d'échelle minuscule a besoin d'un grand poids pour peser : Ridge la punit injustement. Le Pipeline de la Séance 4 s'impose : `StandardScaler` puis `Ridge`, le scaler ajusté sur le train seulement.

Troisième rendez-vous de la standardisation : le kNN (distances), la descente de gradient (cuvette ronde), Ridge (pénalité équitable). Ce n'est pas un détail de cuisine — c'est une pièce du contrat de validité des modèles.

Une variante : Lasso (et Elastic Net)

L'idée en une phrase

Lasso suit la même idée que Ridge — pénaliser la taille des coefficients — mais sur la *somme des valeurs absolues* au lieu des carrés. Conséquence frappante : là où Ridge *rétrécit* sans jamais annuler, Lasso pousse certains coefficients **exactement à zéro** et fait donc de la **sélection de variables**.

$$J_{\text{lasso}} = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^d |w_j| \quad (\text{Ridge : } \alpha \sum_j w_j^2)$$

Image : Ridge *baisse les salaires de tous*, Lasso *licencie les moins utiles*. On prend Ridge quand toutes les variables comptent un peu (stabiliser le modèle), Lasso quand on en soupçonne beaucoup d'inutiles (modèle simple, interprétable). Un compromis, l'*Elastic Net*, combine les deux pénalités. Les trois s'emploient comme [LinearRegression](#), dans un Pipeline après le [StandardScaler](#), et se règlent par α en validation (le creux du U).

Synthèse

Le protocole de régression

De bout en bout

1. Cible continue identifiée ($y \in \mathbb{R}$, en jours) → **2.** split train/test → **3.** Pipeline (imputation, standardisation si descente ou Ridge) → **4.** `LinearRegression` — ou `Ridge`, ou `SGDRegressor` selon l'échelle du problème → **5.** MAE + RMSE + R^2 *sur le test*, comparés à la baseline-moyenne → **6.** graphe des résidus → **7.** si complexité à régler (degré, α) : choisir au creux du U de *validation*, jamais au train.

À retenir — Séance 5

L'essentiel

- La régression prédit un **réel** ; le modèle linéaire le fait par somme pondérée : $\hat{y} = b + \mathbf{w} \cdot \mathbf{x}$, et ses coefficients se lisent en clair.
- La **MSE** définit la meilleure droite ; sa cuvette admet un minimum **exact** : l'équation normale $\hat{\theta} = (\tilde{X}^\top \tilde{X})^{-1} \tilde{X}^\top y$.
- La **descente de gradient** atteint le même minimum par pas successifs $\theta \leftarrow \theta - \eta \nabla J$; le pas η arbitre entre lenteur et divergence. C'est l'algorithme d'apprentissage universel du ML.
- **MAE** et **RMSE** parlent en jours (la RMSE surtaxe les grandes erreurs) ; le **R²** compare à la baseline-moyenne.
- Le degré polynomial est un bouton de complexité : le train flatte, la **validation** tranche (courbe en U).
- **Ridge** ($+\alpha \sum w_j^2$) rétrécit tous les coefficients ; **Lasso** ($+\alpha \sum |w_j|$) en annule certains et sélectionne les variables. Standardiser d'abord, régler α en validation.

Pièges fréquents

- Lire les coefficients de variables d'échelles différentes comme des importances comparables.
- Confondre association et causalité dans un poids w_j .
- Annoncer un R^2 sans MAE/RMSE — ou l'inverse : les deux familles se complètent.
- Choisir degré ou α sur l'erreur d'*entraînement* : elle vote toujours pour la complexité maximale.
- Lancer une descente de gradient (ou Ridge) sur des variables non standardisées.
- Oublier que le prétraitement s'ajuste sur le train seul — la fuite de la Séance 4 guette aussi la régression.
- Un R^2 négatif sur le test n'est pas un bug de `scikit-learn` : le modèle fait pire que la moyenne.

Prochaine séance

Séance 6 — Classifier et choisir ses métriques

Retour aux classes, mais en profondeur. Le modèle linéaire d'aujourd'hui en est le **cœur** : la *même* somme pondérée $z = b + \mathbf{w} \cdot \mathbf{x}$, cette fois passée dans une **sigmoïde** pour devenir une probabilité — c'est la régression *logistique*. Puis la matrice de confusion, precision, recall, F1, les courbes ROC, et le cas qui fâche : prédire le paludisme, présent chez 11,8 % des patients seulement, là où l'accuracy devient une métrique menteuse.

Références

- James, Witten, Hastie, Tibshirani — *An Introduction to Statistical Learning*, 2^e éd., Springer, 2021.
- Géron — *Hands-On Machine Learning*, 3^e éd., O'Reilly, 2022.
- Müller, Guido — *Introduction to Machine Learning with Python*, O'Reilly, 2016.
- Cours : University of Washington CSE 446 ; Stanford CS229.

Ressources de la séance

- **Notebook** · [TP 5 — Régression](#) (647 Ko)
- **TD** · [Fiche d'exercices 5](#) (184 Ko)