Fiche de révision – PC1

La régression linéaire

Oulad Ali Ayoub

TD X-HEC · APM_51434_EP · École Polytechnique

Table des matières

1 La régression linéaire en une image . . . 2
2 Le cadre : données, classe de fonctions . . . 3
3 La régression linéaire : le problème . . . 4
4 Convexité et existence de la solution . . . 5
4.1 Ensemble convexe, fonction convexe . . . 5
4.2 La solution existe toujours . . . 5
5 Comment montrer qu'une fonction est convexe ? . . . 6
5.1 La Hessienne . . . 6
5.2 Gradient nul = optimum (si convexe) . . . 7
6 À connaître : gradient et formule de $\hat{w}$ . . . 8
6.1 Le gradient de la perte . . . 8
6.2 La formule de $\hat{w}$ . . . 9
6.3 Quand X n'est pas injective : la régularisation . . . 10
7 Le modèle statistique : MCO (OLS) . . . 12
7.1 Le modèle et l'estimateur . . . 12
7.2 Trois définitions à connaître . . . 12
7.3 Propriétés de l'estimateur OLS (voir exercice 1.4) . . . 13
7.4 L'estimateur GLS (voir exercice 1.5) . . . 13
8 La régression polynomiale et la matrice de design . . . 14
8.1 Dans le TP : choisir le degré . . . 16
9 Autotest : 16 questions éclair . . . 18

Le fil de la fiche

1. La régression linéaire en une image : on minimise des **carrés**.
2. Le cadre : données $\mathcal{D}$, classe de fonctions $\mathcal{F} \subset \mathcal{H}$ (ni trop petite, ni trop grande).
3. Le problème : trouver $\hat{w}$; l'objectif à minimiser.
4. Convexe (ensemble, fonction), existence de $\hat{w}$, Hessienne, gradient nul.
5. À connaître : le gradient de L et la formule de $\hat{w}$; en pratique avec NumPy.
6. Le modèle $y = Xw^\star + \varepsilon$: MCO (OLS), sans biais, variance, Gauss–Markov ; puis GLS.
7. La régression polynomiale et la matrice de design.

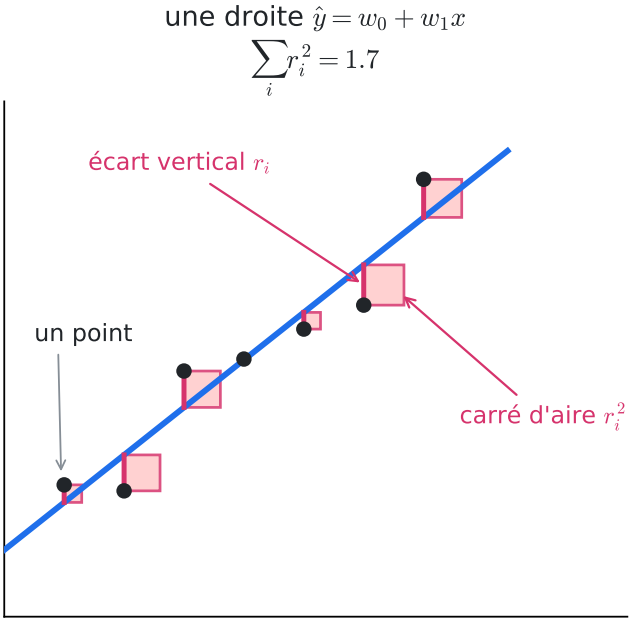
Comment lire

Chaque page : **un dessin + l'essentiel**. Boîte bleue = vie de tous les jours ; verte = à retenir ; rose = piège ; orange = réflexe. *Les exercices se corrigent ensemble en séance : ici, uniquement ce qu'il faut savoir avant.*

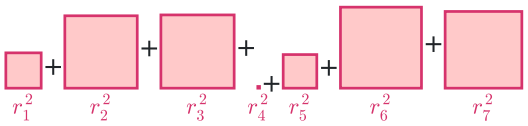
1 La régression linéaire en une image

Dans la vraie vie

Tu as des points (x_i, y_i) (surface d'un appartement, loyer) et tu veux **tracer une droite** qui passe « au plus près ». « Au plus près » = pour chaque point, tu mesures l'**écart vertical** à la droite, tu dessines un **carré** de ce côté, et tu **additionnes les aires**. La meilleure droite est celle qui rend cette somme la plus petite.



on additionne toutes les aires



$$L(w) = \tfrac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \tfrac{1}{2}\|Xw - y\|^2$$

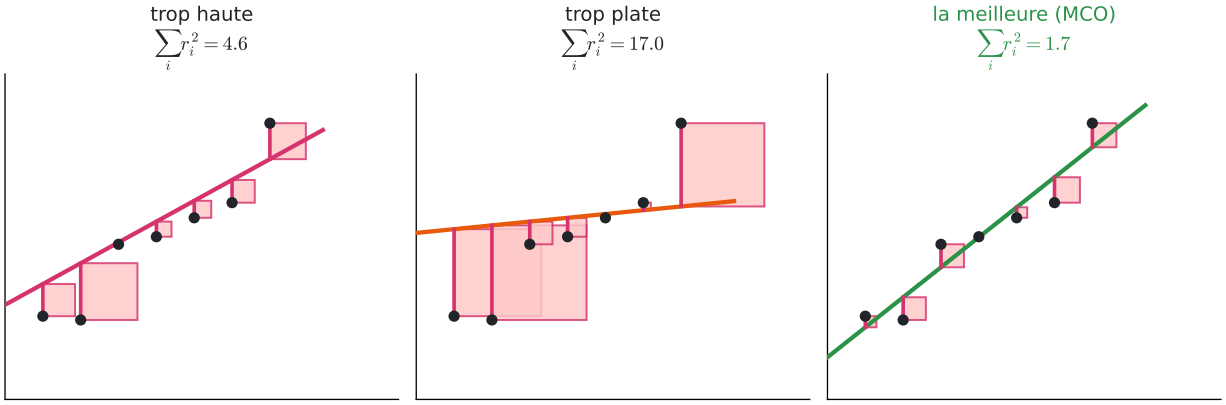
on cherche la droite qui rend cette somme la plus petite

À retenir

On ne minimise pas directement les écarts : on minimise la **somme des carrés** des écarts

$$L(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad \text{avec} \qquad \hat{y}_i = w_0 + w_1 x_i.$$

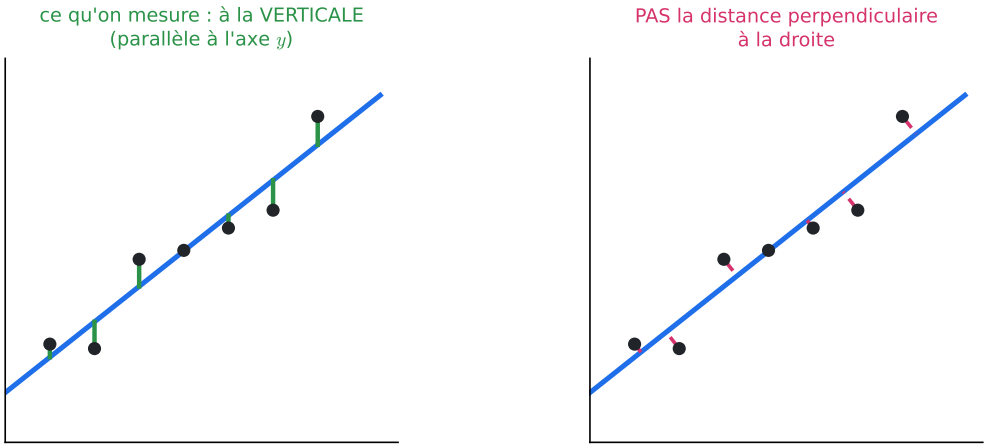
Un écart de 2 compte 4, un écart de 3 compte 9 : les **gros écarts sont très punis**.



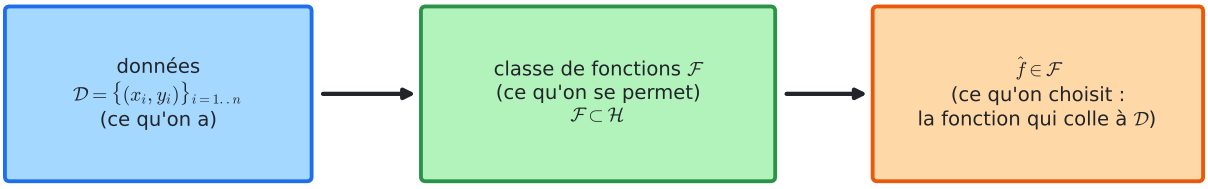
Même données, trois droites : plus la droite s'éloigne des points, plus les carrés sont gros. La meilleure droite (verte) a la plus petite somme.

Quel écart ?

L'écart se mesure **à la verticale** (parallèlement à l'axe y) : c'est $y_i - \hat{y}_i$. Ce n'est **pas** la distance perpendiculaire à la droite.



2 Le cadre : données, classe de fonctions

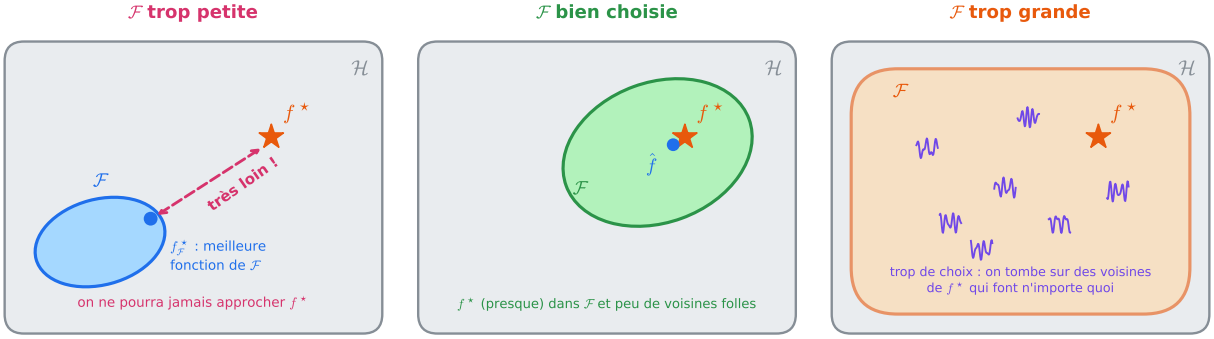


$\mathcal{H}$ = l'ensemble de TOUTES les fonctions possibles ; $f^\star$ = la vraie fonction, inconnue, qu'on voudrait trouver

Les trois ingrédients

- un **jeu de données** $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$: ce qu'on observe ;
- $\mathcal{H}$: l'ensemble de **toutes** les fonctions possibles ; $f^\star \in \mathcal{H}$ est la **vraie fonction**, inconnue, qu'on voudrait trouver ;
- une **classe de fonctions** $\mathcal{F} \subset \mathcal{H}$: celles qu'on s'autorise à essayer. On choisit $\hat{f} \in \mathcal{F}$ qui colle le mieux aux données $\mathcal{D}$.

Pour la régression linéaire : $\mathcal{F} = \{x \mapsto w^\top x \ : \ w \in \mathbb{R}^d\}$.



	Ce qui se passe	Conséquence
$\mathcal{F}$ **trop petite**	$f^\star \notin \mathcal{F}$. La meilleure fonction de $\mathcal{F}$ (la **projection** $f^\star_{\mathcal{F}}$ de $f^\star$ sur $\mathcal{F}$) peut être **très loin** de $f^\star$	on n'approchera jamais $f^\star$, même avec *énormément* de données
$\mathcal{F}$ **trop grande**	elle contient $f^\star$, mais aussi **plein de fonctions voisines** qui collent aux données et font **n'importe quoi** ailleurs	on met longtemps à trouver la bonne, et on peut tomber sur une mauvaise
$\mathcal{F}$ **bien choisie**	assez riche pour approcher $f^\star$, assez petite pour qu'on la repère	c'est le but

Le but

Trouver une **bonne famille** $\mathcal{F}$: ni trop pauvre (on rate $f^\star$), ni trop riche (on se perd). La régression polynomiale (section 8) en est l'exemple parfait : plus le degré monte, plus la famille grandit.

3 La régression linéaire : le problème

Définition

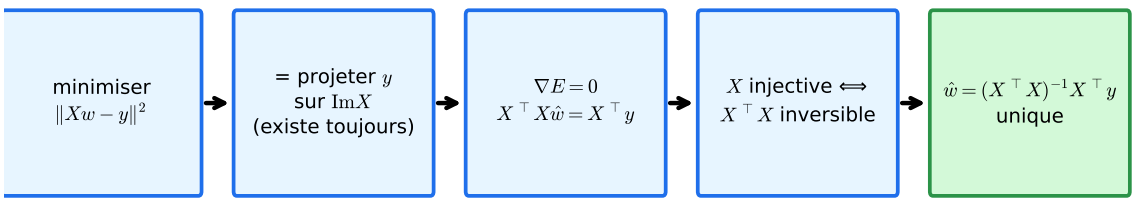
Données : $X \in \mathbb{R}^{n\times d}$ (n lignes = n observations, d colonnes = d variables) et $y \in \mathbb{R}^n$. **Faire une régression linéaire**, c'est trouver le vecteur $\hat{w} \in \mathbb{R}^d$ tel que $X\hat{w}$ soit le plus proche possible de y :

$$\hat{w} \in \arg\min_{w\in\mathbb{R}^d} L(w), \qquad L(w) = \frac{1}{2}\|Xw - y\|_2^2 = \frac{1}{2}\sum_{i=1}^{n}\left(x_i^\top w - y_i\right)^2.$$

La prédiction pour un nouveau point x est $\hat{f}(x) = \hat{w}^\top x$.

Piège classique

arg min est un **ensemble** (les w qui réalisent le minimum), d'où l'écriture « $\hat{w} \in \arg\min$ ». Il peut contenir un seul point ou plusieurs. Le $\frac{1}{2}$ ne change pas le minimiseur : il est là pour simplifier la dérivée.



Le plan de la fiche pour trouver $\hat{w}$: minimiser → projeter → gradient nul → injectivité → formule.

4 Convexité et existence de la solution

4.1 Ensemble convexe, fonction convexe

Ensemble convexe

$C \subset \mathbb{R}^d$ est **convexe** si, pour tous $x, y \in C$ et tout $t \in [0,1]$:

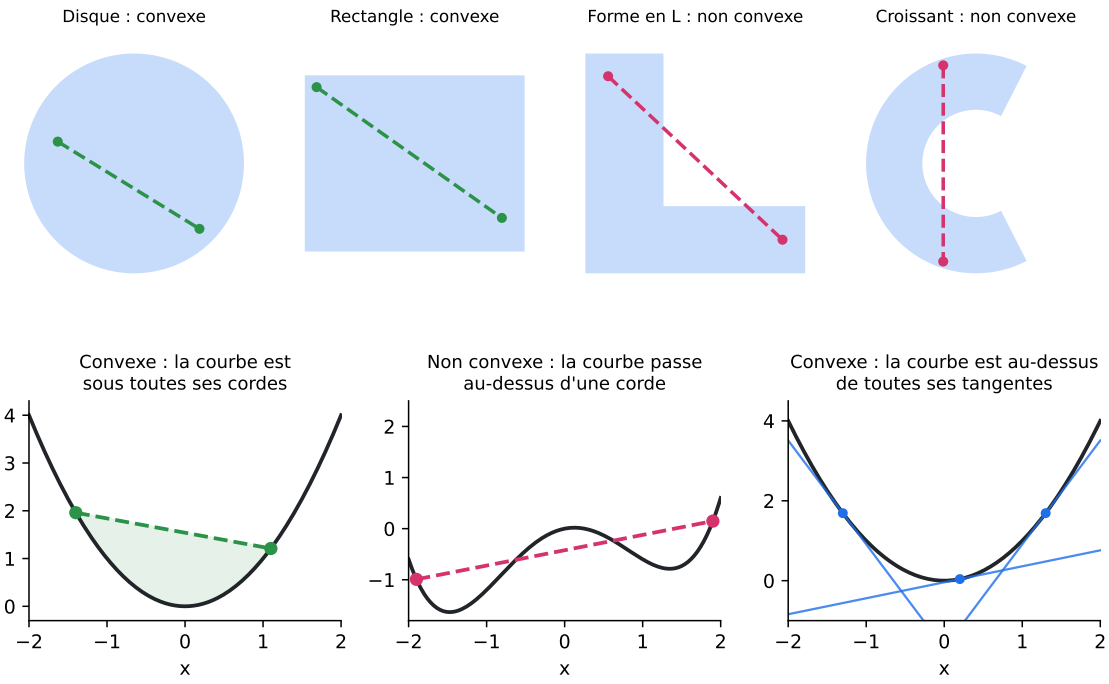
$$tx + (1-t)y \in C \qquad \text{(le segment } [x,y] \text{ reste dans } C\text{)}.$$

Fonction convexe

Soit C convexe et $f : C \to \mathbb{R}$. f est **convexe** si, pour tous $x, y \in C$ et tout $t \in [0,1]$:

$$f\big(tx + (1-t)y\big) \;\leq\; t\,f(x) + (1-t)\,f(y) \qquad \text{(la courbe est sous ses cordes)}.$$

f est **strictement convexe** si l'inégalité est **stricte** pour $x \neq y$ et $t \in]0,1[$.



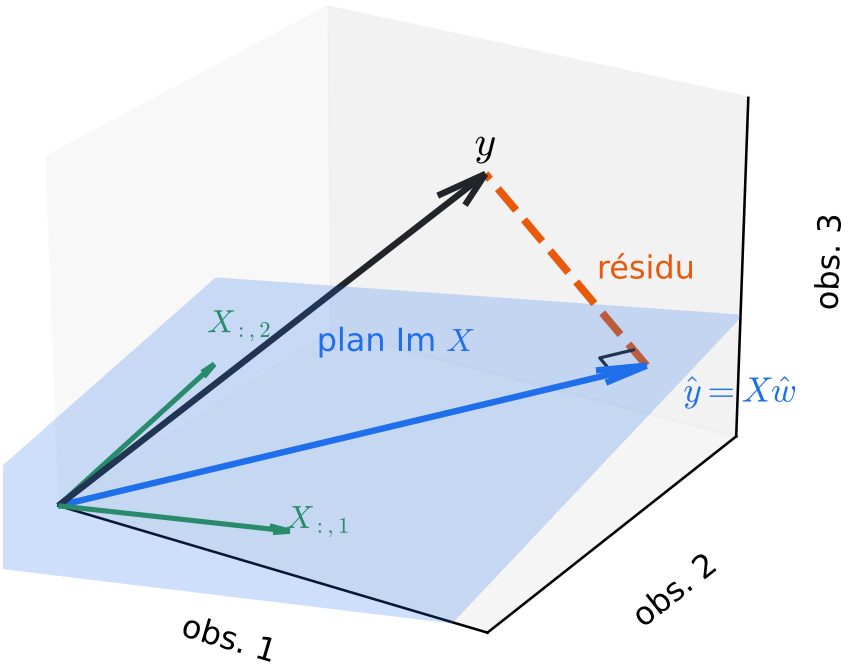
Ensemble $\neq$ fonction

« Convexe » ne veut pas dire la même chose : pour un **ensemble**, le segment reste **dedans** ; pour une **fonction**, la corde reste **au-dessus** de la courbe.

4.2 La solution existe toujours

Propriété

Pour tout $X \in \mathbb{R}^{n \times d}$ et tout $y \in \mathbb{R}^n$: $\arg\min_w \|Xw - y\|_2 \neq \emptyset$, c'est-à-dire qu'il existe **toujours** un $\hat{w}$.
Pourquoi ? $X\hat{w}$ est la **projection orthogonale** de y sur le sous-espace $\operatorname{Im} X = \{Xw : w \in \mathbb{R}^d\}$. Cette projection existe toujours, et comme elle est dans $\operatorname{Im} X$, elle s'écrit $X\hat{w}$ pour un certain $\hat{w}$.



Existence $\neq$ unicité

$\hat{y} = X\hat{w}$ (la projection) est **unique**. Mais $\hat{w}$ n'est unique que si X est **injective** (colonnes indépendantes). Colonnes redondantes $\Rightarrow$ plusieurs $\hat{w}$ donnent la même projection.

5 Comment montrer qu'une fonction est convexe ?

5.1 La Hessienne

Notation

Si $f : \mathbb{R}^d \to \mathbb{R}$ est deux fois différentiable, sa **Hessienne** en x est la matrice des dérivées secondes :

$$H_f(x) = \nabla^2 f(x) = \left(\frac{\partial^2 f}{\partial x_i \, \partial x_j}(x)\right)_{1 \le i,j \le d} \in \mathbb{R}^{d \times d}.$$

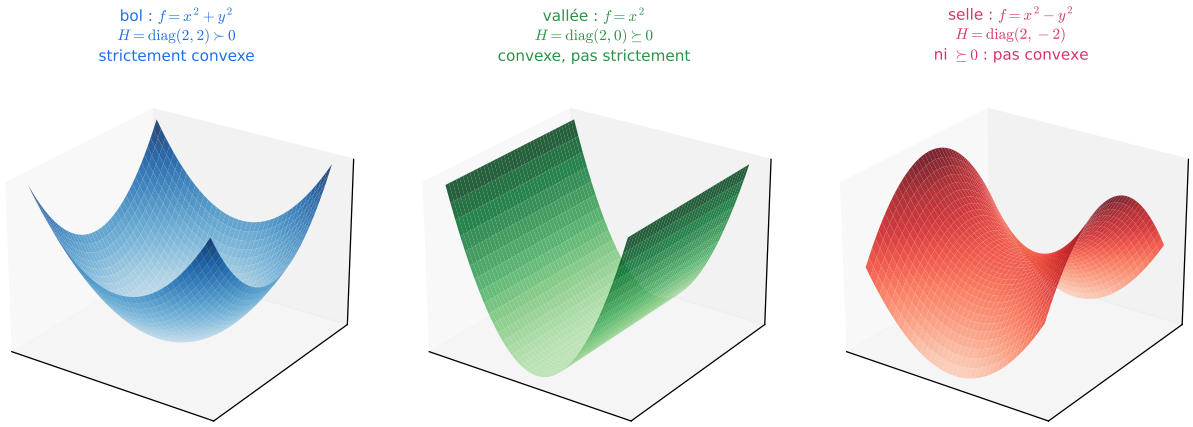
*Ici $\nabla^2 f$ veut dire « le gradient du gradient » : $H_f = \nabla^2 f$ est une **matrice** ; on utilise les deux notations indifféremment. Elle est symétrique.*

Le critère

Si f est deux fois différentiable sur $\mathbb{R}^d$:

- $H_f(x) \succeq 0$ pour tout $x \iff f$ est **convexe** ;
- $H_f(x) \succ 0$ pour tout $x \implies f$ est **strictement convexe**.

Rappel : A symétrique est $\succeq 0$ (*semi-définie positive*) si $v^\top A v \geq 0$ pour tout v ; elle est $\succ 0$ (*définie positive*) si $v^\top A v > 0$ pour tout $v \neq 0$.



Piège classique

La réciproque du « $\succ 0$ » est fausse : $f(x) = x^4$ est strictement convexe mais $f''(0) = 0$. Et en dimension 1 la Hessienne n'est que $f''(x)$: le critère devient « $f'' \geq 0$ ».

Montrer qu'une fonction est convexe

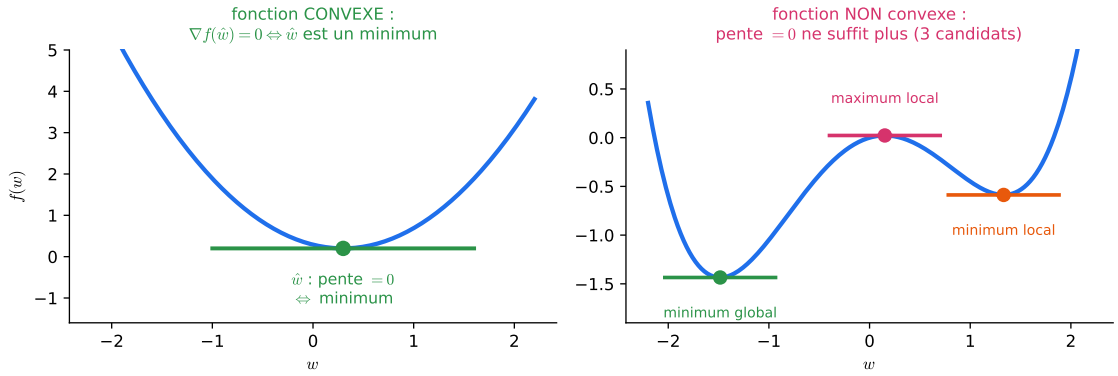
(1) Est-elle deux fois différentiable ? (2) Calcule sa Hessienne. (3) Montre qu'elle est $\succeq 0$ (en écrivant $v^\top H v$ comme une somme de carrés, par exemple). Si tu trouves $\succ 0$, elle est même strictement convexe.

5.2 Gradient nul = optimum (si convexe)

À connaître

Soit $f : \mathbb{R}^d \to \mathbb{R}$ **convexe** et différentiable (en parenthèses : strictement convexe $\Rightarrow$ le minimiseur est unique). Alors

$$\hat{w} \in \arg\min f \quad (\hat{w} \text{ est optimal}) \qquad \Longleftrightarrow \qquad \nabla f(\hat{w}) = 0.$$



Piège classique

Sans convexité, « gradient nul » ne suffit pas : on peut tomber sur un maximum local, un minimum local ou un point selle. **C'est la convexité qui rend la condition suffisante.**

6 À connaître : gradient et formule de $\hat{w}$

6.1 Le gradient de la perte

La perte, écrite avec des sommes

$$L(w) = \frac{1}{2}\|Xw - y\|^2 = \frac{1}{2}\sum_{i=1}^{n}\Big(\sum_{j=1}^{d} X_{ij}w_j - y_i\Big)^2.$$

À connaître : la dérivée partielle

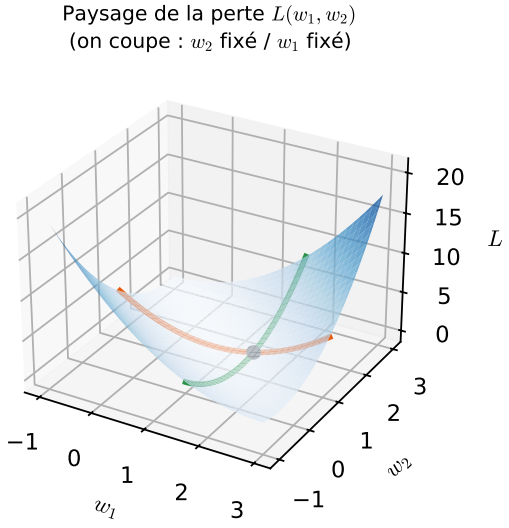
On dérive en w_j (les autres w_k sont des constantes) : seul le terme avec w_j compte.

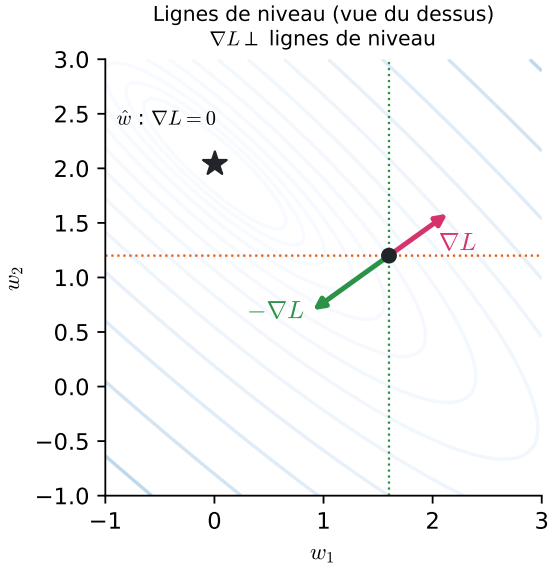
$$\boxed{\frac{\partial L}{\partial w_j}(w) = \sum_{i=1}^{n} X_{ij}\big((Xw)_i - y_i\big) = (X_{:,j})^\top (Xw - y)}$$

= *le produit scalaire de la j-ième colonne de X avec le résidu* $Xw - y$.

Le gradient = toutes les dérivées partielles empilées

$$\nabla L(w) = \begin{pmatrix} \frac{\partial L}{\partial w_1}(w) \\ \vdots \\ \frac{\partial L}{\partial w_d}(w) \end{pmatrix} = \begin{pmatrix} (X_{:,1})^\top (Xw - y) \\ \vdots \\ (X_{:,d})^\top (Xw - y) \end{pmatrix} \implies \boxed{\nabla L(w) = X^\top (Xw - y)}.$$





Équations normales

$$\nabla L(\hat{w}) = 0 \iff X^\top (X\hat{w} - y) = 0 \iff \boxed{X^\top X\hat{w} = X^\top y}.$$

En mots : le résidu $y - X\hat{w}$ est **orthogonal** à toutes les colonnes de X (c'est la projection).

L est convexe

La Hessienne de L est $\nabla^2 L(w) = X^\top X$ (elle ne dépend pas de w). Or $v^\top X^\top X v = \|Xv\|^2 \geq 0$: donc $X^\top X \succeq 0$ et L est **convexe**. Si X est injective, $X^\top X \succ 0$ et L est **strictement convexe**. *C'est pour ça que « gradient nul ⇔ minimum » s'applique ici.*

6.2 La formule de $\hat{w}$

En théorie

On suppose X **injective** (colonnes indépendantes ; il faut $d \leq n$). Alors

$$X \text{ injective} \iff X^\top X \text{ inversible},$$

on peut inverser dans les équations normales et on obtient l'**unique** minimiseur

$$\boxed{\hat{w} = (X^\top X)^{-1} X^\top y}.$$

En pratique

X **n'est pas toujours injective** (colonnes redondantes, $d > n$) : $X^\top X$ n'est alors pas inversible. Et même quand elle l'est, **on ne calcule pas l'inverse** : on utilise une bibliothèque (NumPy).

Fonction NumPy	Ce qu'elle fait
`np.linalg.solve(A, b)`	résout $Ax = b$ sans former l'inverse (précis, rapide) ; exige A inversible
`np.linalg.inv(A)`	calcule A^{-1} : marche, mais **moins précis** ; à éviter
`np.linalg.lstsq(X, y)`	résout directement les moindres carrés ; **marche même si X n'est pas injective** (renvoie la solution de norme minimale)

Exemple de code (5 points, un intercept) : les points $(1,2), (2,3), (3,5), (4,4), (5,6)$.

```
import numpy as np
x = np.array([1., 2., 3., 4., 5.])
y = np.array([2., 3., 5., 4., 6.])
X = np.column_stack([np.ones(5), x])      # colonne de 1 + colonne x   (X est 5x2)

w1 = np.linalg.solve(X.T @ X, X.T @ y)           # [1.3 0.9]   on résout X^T X w = X^T y
w2 = np.linalg.inv(X.T @ X) @ X.T @ y            # [1.3 0.9]   marche, mais déconseillé
w3 = np.linalg.lstsq(X, y, rcond=None)[0]        # [1.3 0.9]   le plus robuste

# X non injective : on ajoute une colonne 2*x (redondante)
X2 = np.column_stack([np.ones(5), x, 2*x])
# np.linalg.solve(X2.T @ X2, X2.T @ y)   -> LinAlgError: Singular matrix
np.linalg.lstsq(X2, y, rcond=None)[0]            # [1.3  0.18  0.36]  (une solution parmi d'autres)
```

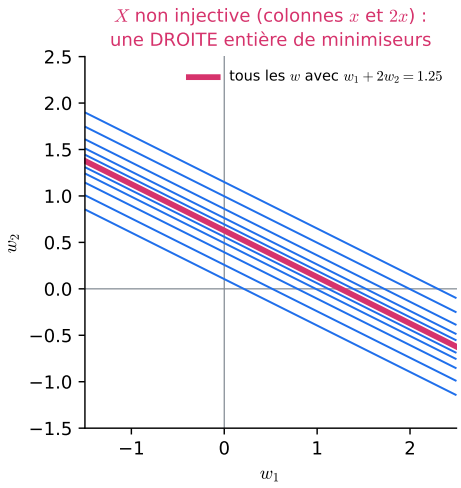
Vérifier ce qu'on obtient

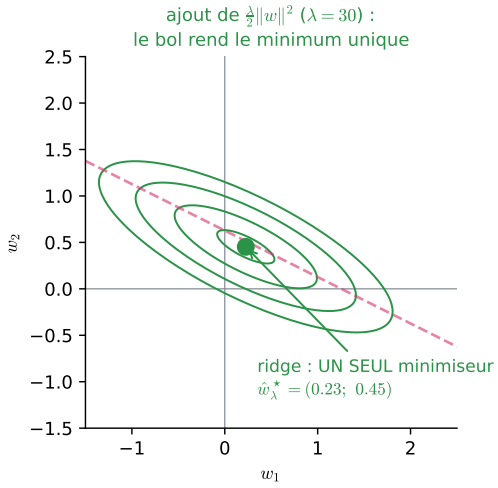
Ici $\hat{w} = (1{,}3;\ 0{,}9)$: la droite $\hat{y} = 1{,}3 + 0{,}9\,x$. À la main : $X^\top X = \begin{pmatrix} 5 & 15 \\ 15 & 55 \end{pmatrix}$, $X^\top y = \begin{pmatrix} 20 \\ 69 \end{pmatrix}$. Dans le cas non injectif, $0{,}18 + 2 \times 0{,}36 = 0{,}9$: le coefficient total sur x est toujours 0,9, seule la *répartition* entre les deux colonnes redondantes n'est pas imposée.

6.3 Quand X n'est pas injective : la régularisation

Idée

Si X n'est pas injective ($X^\top X$ non inversible), la formule $(X^\top X)^{-1}X^\top y$ ne marche plus. On peut alors utiliser des méthodes de **régularisation**, appelées **ridge** et **lasso** : on **ajoute une pénalité** qui décourage les grands coefficients.





Le ridge

On ajoute $\frac{\lambda}{2}\|w\|^2$ à la perte ($\lambda > 0$: le *réglage*) :

$$L_\lambda(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2.$$

Gradient : $\nabla L_\lambda(w) = X^\top(Xw - y) + \lambda w$. **Hessienne** : $X^\top X + \lambda I \succ 0$: **strictement convexe**, donc un **unique** minimiseur.
Gradient nul : $(X^\top X + \lambda I)\, w = X^\top y$, d'où

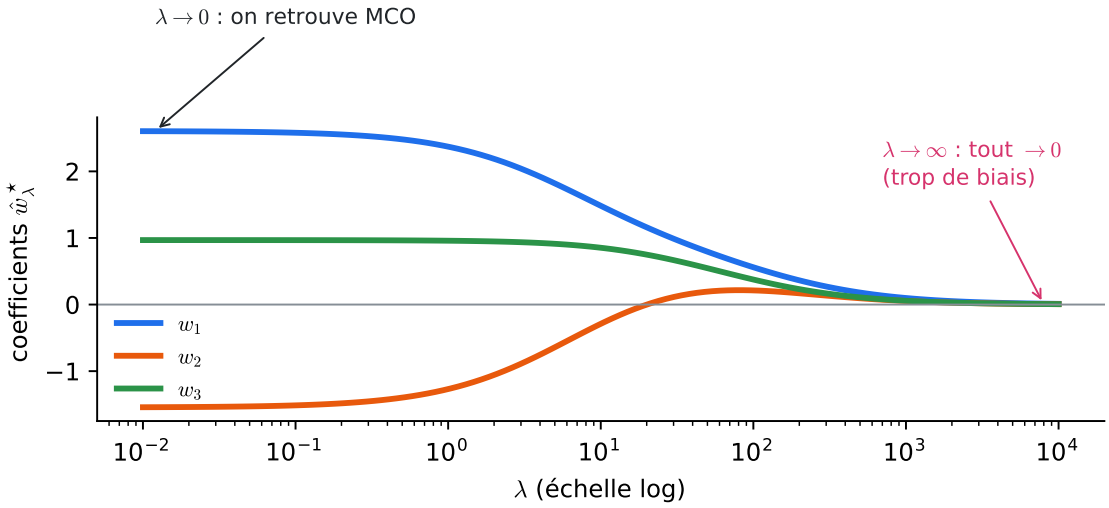
$$\boxed{\hat{w}_\lambda^\star = (X^\top X + \lambda I)^{-1} X^\top y}$$

et **la matrice $X^\top X + \lambda I$ est toujours inversible**, même si X n'est pas injective.

Pourquoi c'est inversible ?

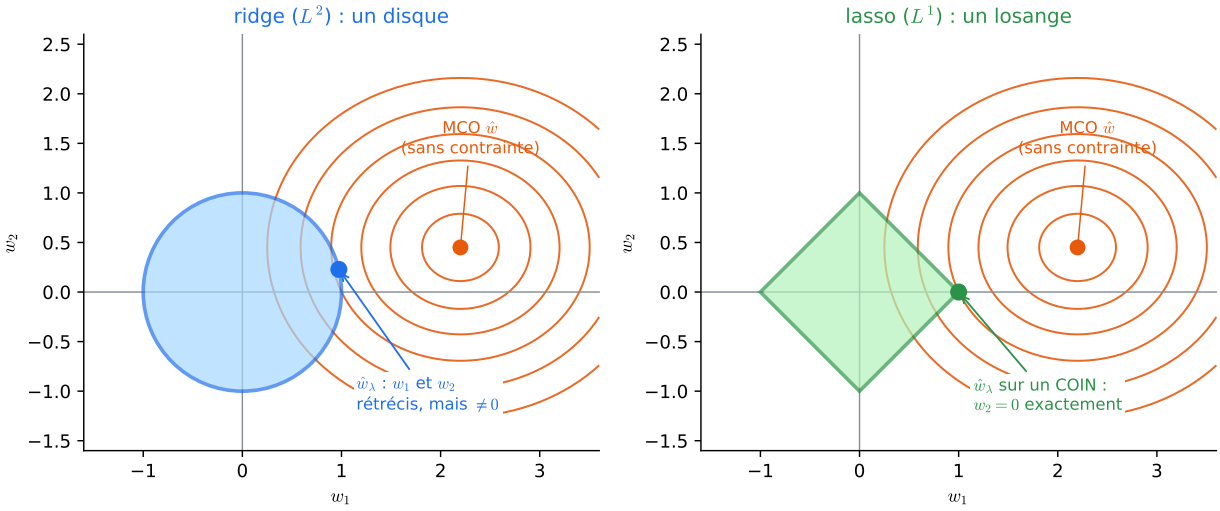
Si $(X^\top X + \lambda I)v = 0$, alors en multipliant par $v^\top$: $\|Xv\|^2 + \lambda\|v\|^2 = 0$, donc $v = 0$. Le noyau est réduit à $\{0\}$.

Valeur de λ	**Effet**
$\lambda \to 0$	on retrouve les moindres carrés $\hat{w}$
λ moyen	coefficients **rétrécis** vers 0, courbe plus sage
$\lambda \to \infty$	$\hat{w}_\lambda^\star \to 0$: le modèle devient trop simple



Le lasso

$L^{\text{lasso}}_\lambda(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1$, avec $\|w\|_1 = \sum_j |w_j|$. **Pas de formule fermée** (la valeur absolue n'est pas dérivable en 0) : on utilise un algorithme. Avantage : il met des coefficients **exactement à** 0.



	Pénalité	Solution	Effet
Ridge	$\frac{\lambda}{2}\|w\|_2^2$	$(X^\top X + \lambda I)^{-1} X^\top y$	rétrécit tout
Lasso	$\lambda\|w\|_1$	pas de formule	annule certains w_j

En code (mêmes données que plus haut, avec la colonne redondante $2x$) :

```
lam = 1.0
d = X2.shape[1]
w_ridge = np.linalg.solve(X2.T @ X2 + lam * np.eye(d), X2.T @ y)
# [0.65  0.21  0.43]   plus de "Singular matrix" ; total sur x : 0.21 + 2*0.43 = 1.07

from sklearn.linear_model import Ridge        # même chose avec scikit-learn
Ridge(alpha=lam).fit(X2[:, 1:], y)            # (alpha = lambda)
```

Détail

`Ridge` de scikit-learn **ne pénalise pas l'intercept** (on lui passe donc les colonnes sans la colonne de 1). Dans la formule ci-dessus, λI pénalise aussi w_0 : c'est un détail, l'idée est la même.

7 Le modèle statistique : MCO (OLS)

7.1 Le modèle et l'estimateur

Le modèle (comme dans le PC1)

$$y = Xw^\star + \varepsilon, \qquad \mathbb{E}\varepsilon = 0, \qquad \operatorname{Var}\varepsilon = \sigma^2 I_n.$$

$w^\star \in \mathbb{R}^d$ est le **vrai** vecteur (inconnu, fixe) ; X est fixe (déterministe) ; **seul** ε **est aléatoire**, donc y l'est aussi.

L'estimateur des moindres carrés ordinaires (MCO / OLS)

Pour X injective :

$$\hat{w}_{\text{OLS}} = (X^\top X)^{-1} X^\top y.$$

C'est exactement la formule de la section précédente, mais vue comme un **estimateur de** $w^\star$: il dépend de y, donc il est aléatoire.

7.2 Trois définitions à connaître

Estimateur

Un **estimateur** de $w^\star$ est une fonction des données (ici de y) : $\tilde{w} = g(y)$. Il est **aléatoire** (il change d'un échantillon à l'autre) ; $w^\star$, lui, est un nombre fixe.

Estimateur linéaire

$\tilde{w}$ est **linéaire** (en y) s'il s'écrit $\tilde{w} = Cy$, où $C \in \mathbb{R}^{d \times n}$ est une **matrice qui ne dépend pas de** y. *L'estimateur OLS est linéaire :* $C = (X^\top X)^{-1} X^\top$.

Estimateur sans biais

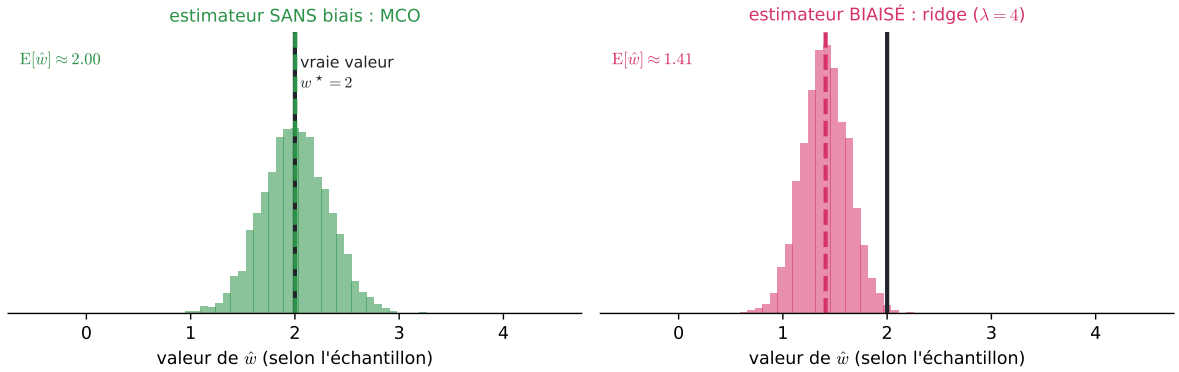
Le **biais** de $\tilde{w}$ est $\mathbb{E}[\tilde{w}] - w^\star$. L'estimateur est **sans biais** si

$$\mathbb{E}[\tilde{w}] = w^\star \qquad \text{(biais nul)}.$$

« Juste **en moyenne** » : une estimation isolée peut être loin, mais sur de très nombreux échantillons la moyenne retombe sur $w^\star$.

Dans la vraie vie

Une balance qui marque **toujours 2 kg de trop** est biaisée. Une balance qui donne 68 puis 73 puis 65 pour la même personne mais **tombe juste en moyenne** est sans biais (mais instable : sa variance est grande).



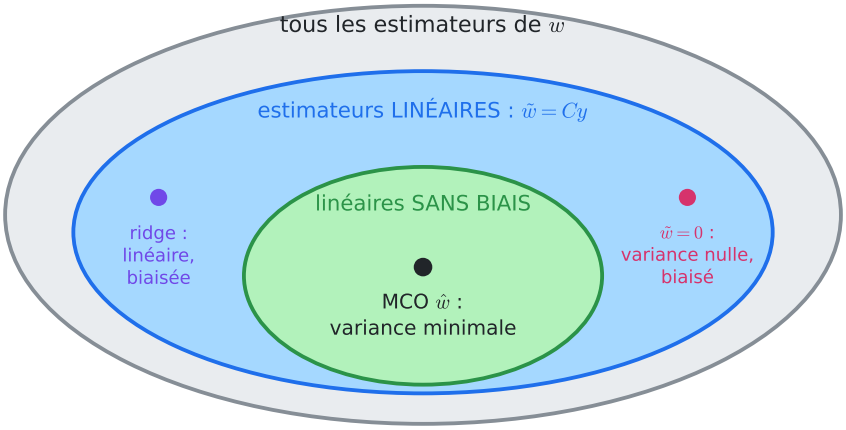
4000 échantillons simulés ($w^\star = 2$). À gauche, les estimations MCO sont centrées sur la vraie valeur. À droite, la ridge ($\lambda = 4$) est plus étroite mais décalée : biaisée.

7.3 Propriétés de l'estimateur OLS (voir exercice 1.4)

Ce qu'il faut savoir

Dans le modèle ci-dessus, avec X injective :

- $\hat{w}_{\mathrm{OLS}}$ est **sans biais** : $\mathbb{E}\hat{w}_{\mathrm{OLS}} = w^\star$;
- sa **variance** (matrice de covariance) est $\mathrm{Var}(\hat{w}_{\mathrm{OLS}}) = \sigma^2 (X^\top X)^{-1}$;
- **Gauss–Markov** : parmi tous les estimateurs **linéaires et sans biais**, c'est celui de **plus petite variance**. Pour tout autre $\tilde{w}$ linéaire sans biais, $\mathrm{Var}(\tilde{w}) - \mathrm{Var}(\hat{w}_{\mathrm{OLS}}) \succeq 0$.



Piège classique

Le « meilleur » ne vaut que parmi les estimateurs **linéaires sans biais** : sans la condition « sans biais », l'estimateur constant $\tilde{w} = 0$ (variance nulle !) gagnerait. La ridge, elle, est biaisée : hors concours. Et il n'est pas nécessaire que ε soit gaussien.

7.4 L'estimateur GLS (voir exercice 1.5)

Quand le bruit n'est plus « égal pour tous »

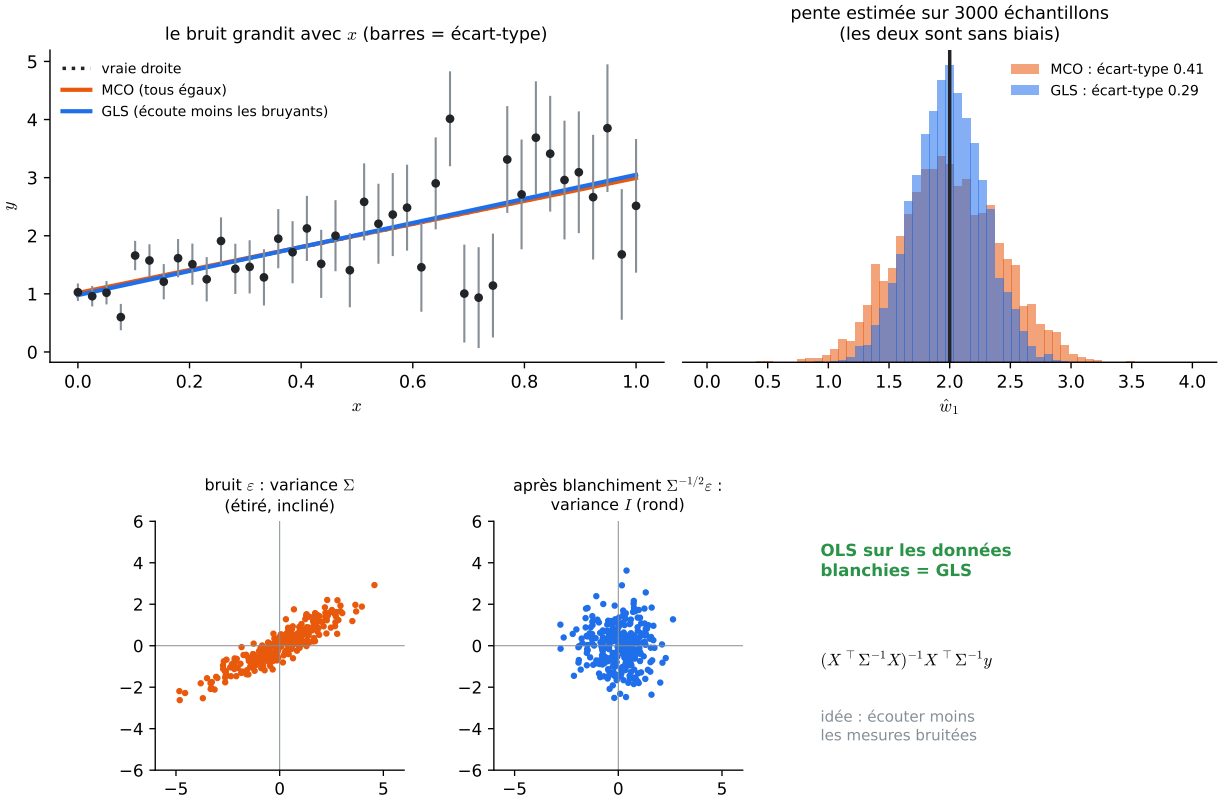
Si $\mathrm{Var}\,\varepsilon = \Sigma \succ 0$ (matrice connue : bruits de tailles différentes, éventuellement corrélés), on définit l'estimateur des moindres carrés généralisés

$$\hat{w}_{\mathrm{GLS}} = (X^\top \Sigma^{-1} X)^{-1} X^\top \Sigma^{-1} y = \arg\min_w (y - Xw)^\top \Sigma^{-1} (y - Xw).$$

On **pondère** : une mesure très bruitée pèse peu. Si $\Sigma = \sigma^2 I$, on retrouve OLS.

Propriété

Quand $\varepsilon \sim \mathcal{N}(0, \Sigma)$ (il suffit en fait de $\mathbb{E}\varepsilon = 0$, $\operatorname{Var}\varepsilon = \Sigma$), $\hat{w}_{\text{GLS}}$ est l'estimateur **linéaire sans biais de plus petite variance**, et $\operatorname{Var}(\hat{w}_{\text{GLS}}) = (X^\top \Sigma^{-1} X)^{-1}$.



GLS = « blanchir » le bruit (pour qu'il redevienne rond, de variance I), puis faire de l'OLS.

8 La régression polynomiale et la matrice de design

Idée

Pour ajuster une **courbe** avec la même machinerie, on **fabrique des colonnes** à partir de x : la matrice de design

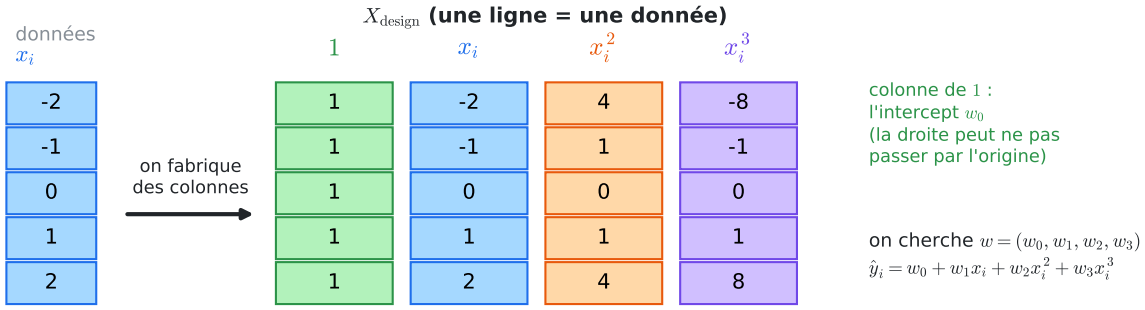
$$X_{\text{design}} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & \cdots & x_2^k \\ \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{pmatrix} \in \mathbb{R}^{n \times (k+1)}.$$

— la **première colonne ne contient que des** 1 : c'est l'**intercept** w_0 (la courbe n'est pas forcée de passer par l'origine) ;

— puis les colonnes $x_i, x_i^2, x_i^3, \ldots, x_i^k$;

— on cherche un vecteur $w = (w_0, \ldots, w_k) : \hat{y}_i = w_0 + w_1 x_i + w_2 x_i^2 + \cdots + w_k x_i^k$.

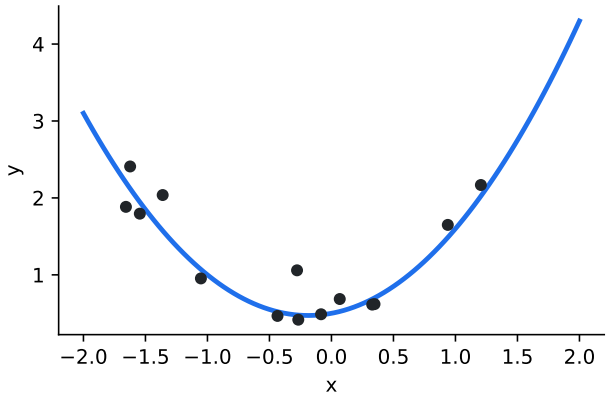
Ensuite **rien ne change** : $\hat{w} = (X_{\text{design}}^\top X_{\text{design}})^{-1} X_{\text{design}}^\top y$.



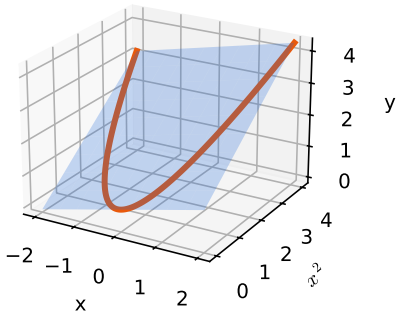
« Linéaire » veut dire linéaire en w

Le modèle est **linéaire en** w (les coefficients), pas en x. C'est ce qui permet de réutiliser tout ce qui précède : minimiser $\|X_{\text{design}} w - y\|^2$.

dans (x, y) : une COURBE



dans (x, x^2, y) : un PLAN

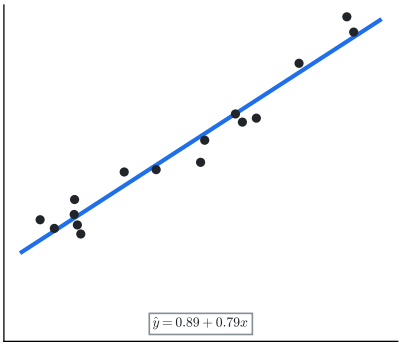


« linéaire » = linéaire en w (les coefficients), pas en x

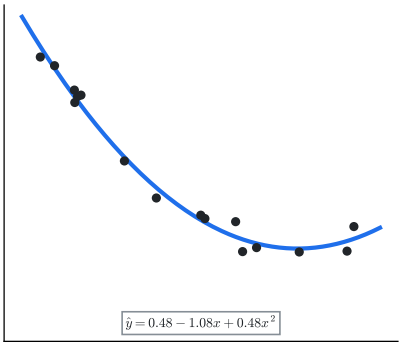
Quelques exemples de colonnes

Colonnes de X	Modèle	Forme
$1,\ x$	$w_0 + w_1 x$	une droite (degré $k = 1$)
$1,\ x,\ x^2$	$w_0 + w_1 x + w_2 x^2$	une parabole ($k = 2$)
$1,\ x,\ x^2,\ x^3$	$w_0 + \cdots + w_3 x^3$	une cubique ($k = 3$)
$1,\ \ln x$	$w_0 + w_1 \ln x$	une courbe en log (toujours linéaire en w)
$1,\ \cos x,\ \sin x$	$w_0 + w_1 \cos x + w_2 \sin x$	une oscillation

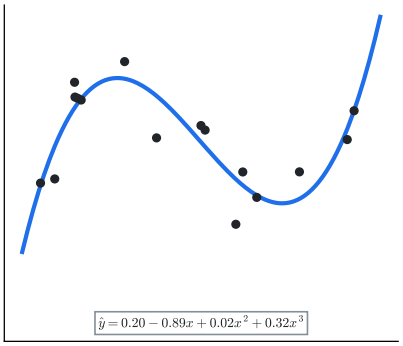
$k = 1$: une droite



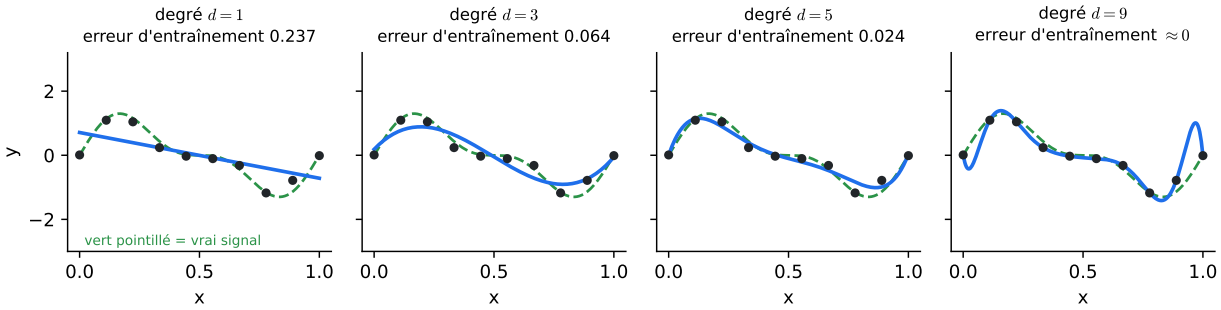
$k = 2$: une parabole



$k = 3$: une cubique

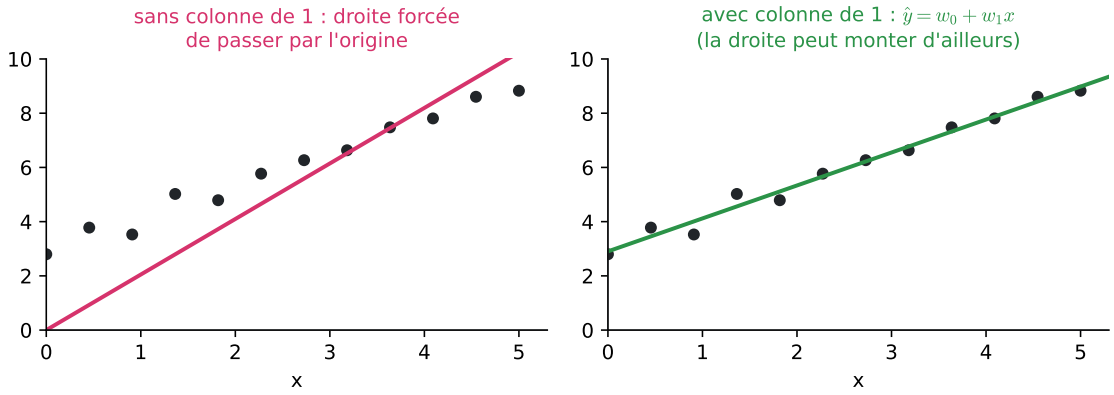


Trois jeux de données, trois degrés : les coefficients sont calculés avec la même formule des moindres carrés.



Retour à la section 2

Les familles $\mathcal{F}_k = \{\text{polynômes de degré} \leq k\}$ sont **emboîtées** : $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \cdots$. Plus k est grand, plus l'erreur sur les données **diminue** (jusqu'à passer par tous les points), mais la courbe se met à osciller entre les points : c'est la famille **trop grande**.

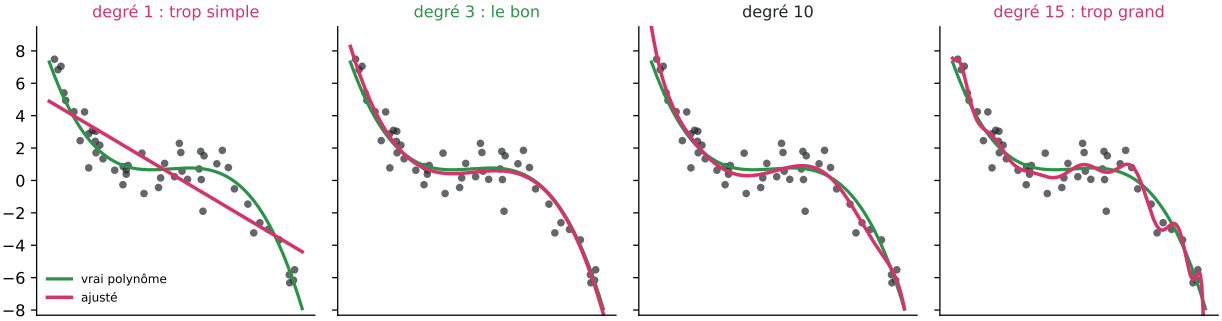


Sans colonne de 1 *(à gauche) la droite est forcée de passer par l'origine ; avec (à droite) elle peut monter ailleurs.*

8.1 Dans le TP : choisir le degré

En code (comme dans le TP) :

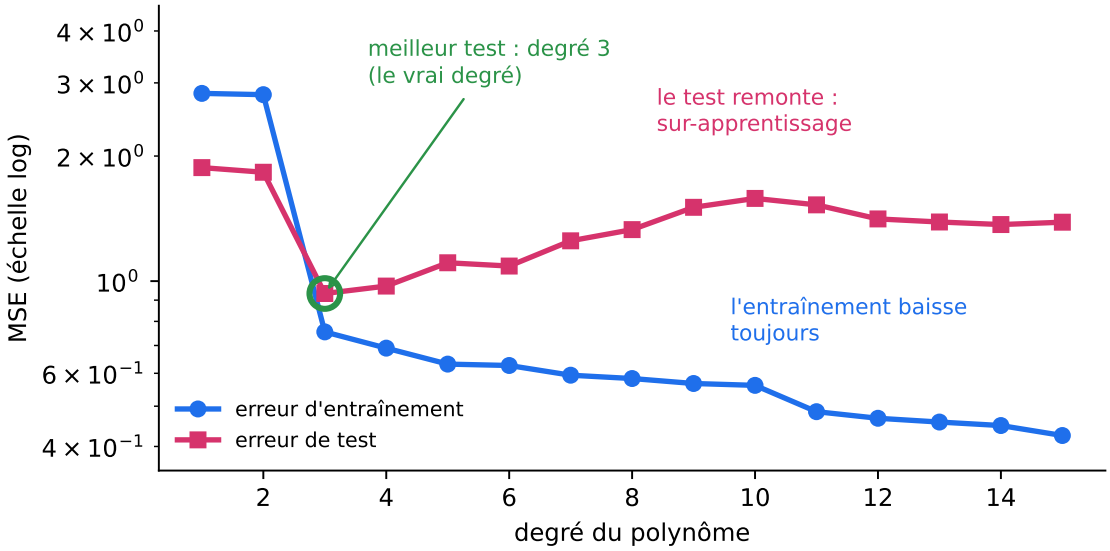
```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree)              # fabrique les colonnes 1, x, x^2, ..., x^degree
X_poly = poly.fit_transform(X)                 # la matrice de design
theta = np.linalg.inv(X_poly.T @ X_poly) @ X_poly.T @ y
y_pred = poly.transform(X_new) @ theta         # prédire en de nouveaux points
```



Mêmes données, quatre degrés : trop simple (1), bon (3), trop souple (10 et 15).

Comment choisir le degré ?

L'erreur **d'entraînement baisse toujours** quand le degré monte : elle ne sert pas à choisir. On regarde l'erreur sur des **données de test**, mises de côté : elle fait un **U**. Le meilleur degré est le fond du U.



Pour aller plus loin

- Avec n petit et un degré élevé, $X^\top X$ devient très mal conditionnée : le **ridge** (section 6.3) stabilise le calcul.
- On peut choisir le degré (et λ) par **validation croisée** : on découpe les données en plusieurs parts, on entraîne sur certaines, on teste sur les autres.
- Les colonnes $x, x^2, \ldots, x^{15}$ n'ont pas la même échelle : on les **standardise** avant un ridge.

9 Autotest : 16 questions éclair

Question	Réponse en une ligne
Qu'est-ce qu'on minimise en régression linéaire ?	la **somme des carrés** des écarts verticaux, $\frac{1}{2}\|Xw - y\|^2$
Que veut dire $\mathcal{F} \subset \mathcal{H}$?	$\mathcal{H}$ = toutes les fonctions ; $\mathcal{F}$ = celles qu'on s'autorise à essayer
$\mathcal{F}$ trop petite ? trop grande ?	trop petite : on ne contient pas $f^\star$; trop grande : plein de voisines folles, on se perd
Ensemble convexe ?	$tx + (1-t)y \in C$ pour tous $x, y \in C$, $t \in [0,1]$
Fonction convexe ?	$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$ (corde au-dessus de la courbe)
Pourquoi $\hat{w}$ existe-t-il toujours ?	$X\hat{w}$ est la projection de y sur Im X
Critère de convexité par la Hessienne ?	$H_f \succeq 0$ partout $\Leftrightarrow$ convexe ; $H_f \succ 0 \Rightarrow$ strictement convexe
Gradient nul : quand est-ce suffisant ?	quand f est **convexe** : $\hat{w}$ optimal $\Leftrightarrow \nabla f(\hat{w}) = 0$
$\nabla L(w)$? Équations normales ?	$X^\top(Xw - y)$; $X^\top X\hat{w} = X^\top y$
Formule de $\hat{w}$?	$(X^\top X)^{-1}X^\top y$ si X injective $\Leftrightarrow$ $X^\top X$ inversible
Et en pratique avec NumPy ?	`np.linalg.solve` ou `lstsq` (pas `inv`) ; `lstsq` marche si X non injective
Estimateur sans biais ? linéaire ?	$\mathbb{E}\tilde{w} = w^\star$; $\tilde{w} = Cy$ avec C fixe
OLS : biais, variance, optimalité ?	sans biais ; $\sigma^2(X^\top X)^{-1}$; variance minimale parmi les linéaires sans biais (Gauss–Markov)
Ridge : fonction, gradient, solution ?	$\frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$; $X^\top(Xw - y) + \lambda w$; $\hat{w}^\star_\lambda = (X^\top X + \lambda I)^{-1}X^\top y$
Ridge vs lasso ?	ridge : toujours inversible, rétrécit ; lasso : pas de formule, annule des coefficients
GLS et régression polynomiale ?	GLS $= (X^\top\Sigma^{-1}X)^{-1}X^\top\Sigma^{-1}y$; polynomiale : colonnes $1, x, x^2, \ldots, x^k$, linéaire en w