S1-MAJEUR - Apprentissage Automatique

Chapitre 3 - Regression linéaire, multilinéaire, logistique

3.1 Regression linéaire et multilinéaire

L'apprentissage s'opère à partir de données exemples

- Ces données exemples reflètent une expérience antérieure, fondée sur des observations.
- Elles incarnent les entités clés du problème, telles que des patients ou des films.
- Chaque donnée (instance) est définie par d valeurs (caractéristiques) ($x = \{x^{(1)}, x^{(2)}, ..., x^{(d)}\}, x \in X \subset \mathbb{R}^d$).
- Les caractéristiques fournissent une représentation des objets, traduisant fréquemment des propriétés pertinentes.
- Un ensemble d'apprentissage se compose de n données exemples ($X = \{x_i \in X, i = 1, ..., n\}$).

L'apprentissage vise à estimer une relation inconnue

- Les données sont liées à une variable cible que l'on cherche à prédire.
- On suppose l'existence d'une relation (inconnue) : $f : X \to Y$, où Y est le domaine de la variable cible Y.
- L'apprentissage vise à estimer f avec une fonction de prédiction (ou modèle) $h : \hat{y} = h(x), \forall x \in X$, où $\hat{y}$ est la prédiction de Y pour x.
- La fonction h appartient à un espace d'hypothèses H, par exemple, l'ensemble des fonctions polynomiales.
- L'apprentissage supervisé signifie que les données d'apprentissage sont annotées, c'est-à-dire associées à leur "vraie" valeur y. Notation : $D = \{(x_i, y_i) \in X \times Y, i = 1, ..., n\}$.

Deux catégories de tâches en apprentissage supervisé

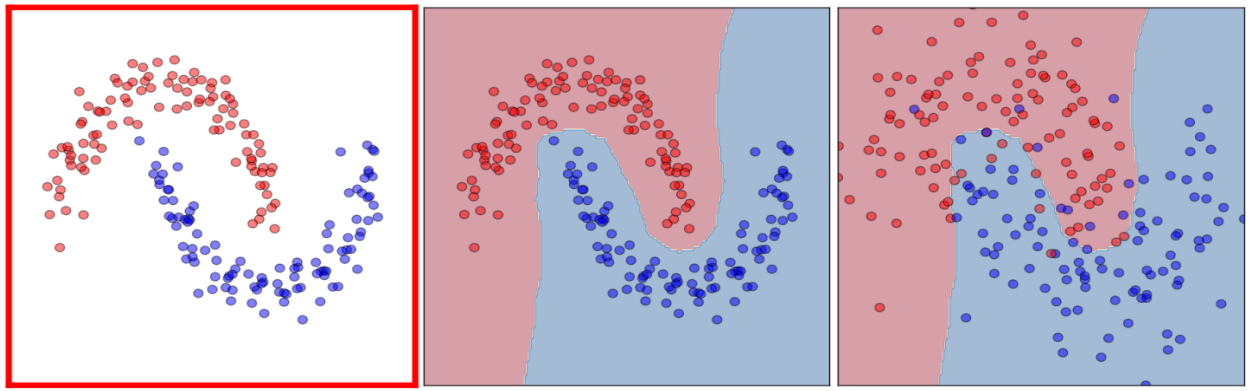
1. **Classification :** La variable cible Y se compose d'un ensemble discret de c valeurs/modalités : $Y = \{\lambda_1, \lambda_2, ..., \lambda_c\}$, avec $c \geq 2$. La fonction h est désignée comme un classifieur.
2. **Régression :** La variable cible Y représente un ensemble continu de nombres réels : $Y \subset \mathbb{R}$. La fonction h est qualifiée de régresseur.

Composition de l'ensemble d'apprentissage

1. **Ensemble d'apprentissage :** Composé de n instances étiquetées où la vraie classe est connue : $D = \{(x_i, y_i) \in X \times Y, i = 1, ..., n\}$.

2. **Modèle appris :** h est appris sur ces données, où $Y = \{rouge, bleu\}$ et $h : X \to Y$.

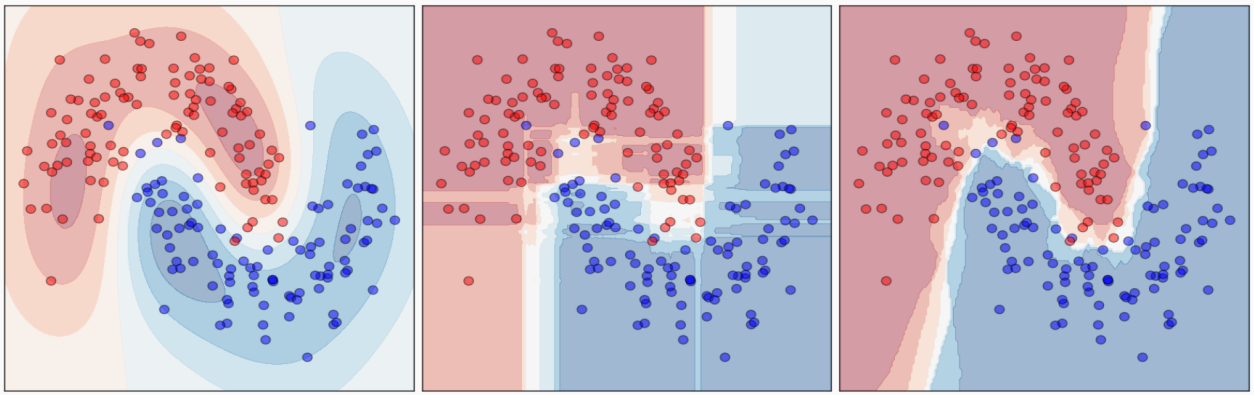
3. **Prédiction :** h prédit "rouge" ou "bleu" pour chaque nouvelle instance $\hat{y} = h(x), \forall x \in \bar{D}$.



Recherche du meilleur modèle

Trouver le meilleur modèle h^* dans l'espace des hypothèses H en minimisant l'erreur de prédiction :

$$h^* = \arg\min_{h \in H} \Pr_{X,Y}(h(x) \neq y)$$



SVM (RBF kernel), Forêts Aléatoires (100 arbres), K-Plus Proches Voisins (K=5)

Optimisation du modèle

L'enjeu des méthodes d'apprentissage est de trouver le meilleur modèle h possible.

- Pour cela, une fonction de perte $L(Y, h(X))$ est introduite pour évaluer la proximité de la prédiction $h(x)$ par rapport à la vraie valeur y:

$$L(y, h(x)) = \begin{cases} 0, & \text{si } h(x) = y \\ \geq 0, & \text{sinon} \end{cases}$$

- L'objectif est de minimiser le risque réel (ou l'erreur en généralisation):

$$R(h) = \mathbb{E}_{X,Y}[L(Y, h(X))]$$

- On cherche à déterminer le modèle h^* qui minimise ce risque:

$$h^* = \arg\min_{h \in H} R(h)$$

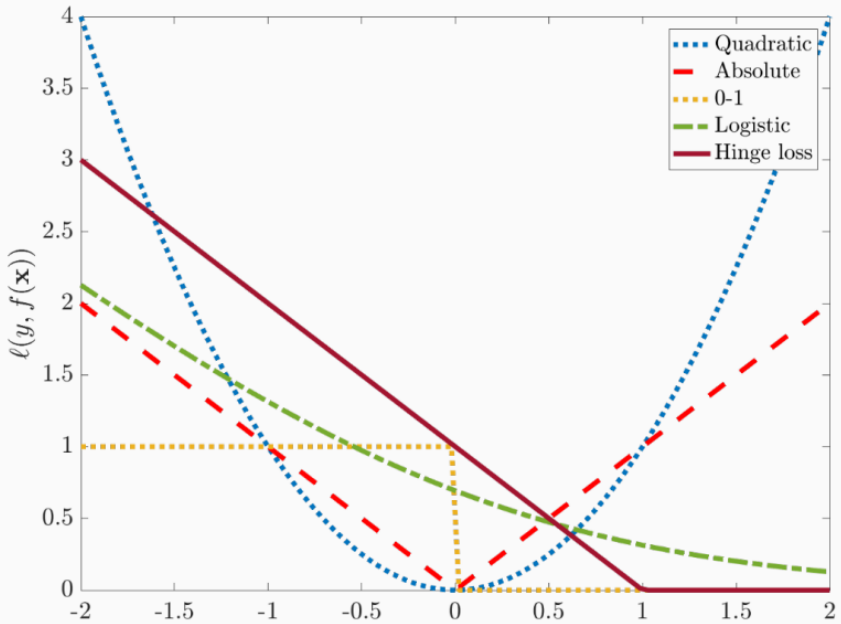
Fonctions de perte usuelles

Quelques fonctions de perte couramment utilisées :

- **Quadratic loss (moindres carrés):** $L(y, h(x)) = (y - h(x))^2$
- L^1 **loss (déviation absolue):** $L(y, h(x)) = |y - h(x)|$
- $0 - 1$ **loss:**

$$L(y, h(x)) = \begin{cases} 0, & \text{si } h(x) = y \\ 1, & \text{sinon} \end{cases}$$

- **Hinge loss:** $L(y, h(x)) = \max(0, 1 - yh(x))$



Estimation du Risque Réel et Risque Empirique

En pratique, le calcul du risque réel ($R(h)$) est impossible car la distribution jointe $P(X, Y)$ est inconnue.

- Nous disposons seulement des données d'apprentissage D pour atteindre notre objectif.
- Le risque réel peut être remplacé par le risque empirique (R_{emp}), une estimation basée sur les données d'entraînement.
- Le risque empirique mesure la performance du modèle sur l'ensemble d'apprentissage, mais il peut ne pas être précis pour de nouvelles données (risque de généralisation).
- Des techniques comme la validation croisée peuvent améliorer cette estimation en évaluant le modèle sur des ensembles de données différents de celui d'entraînement

Estimation du Risque Réel et Risque Empirique

Nous désignerons l'erreur ou, le risque empirique à évaluer par la formule suivante :

$$R_{\text{emp}}(h) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, h(x_i))$$

Et là encore, nous cherchons à minimiser le risque empirique pour trouver le meilleur modèle h^*:

$$h^* = \arg\min_{h \in H} R_{\text{emp}}(h)$$

Cependant, cela peut être difficile en cas de sur-apprentissage, notamment si H est très grand (Il existe plusieurs formules ou modèles et il faut trouver un équilibre : le modèle doit être assez complexe pour saisir la structure sous-jacente des données, mais pas tellement complexe qu'il mémorise le bruit, d'où l'intérêt de faire de la validation croisée), n est petit, ou si D n'est pas assez représentatif du problème.

Sur-apprentissage et Sous-apprentissage

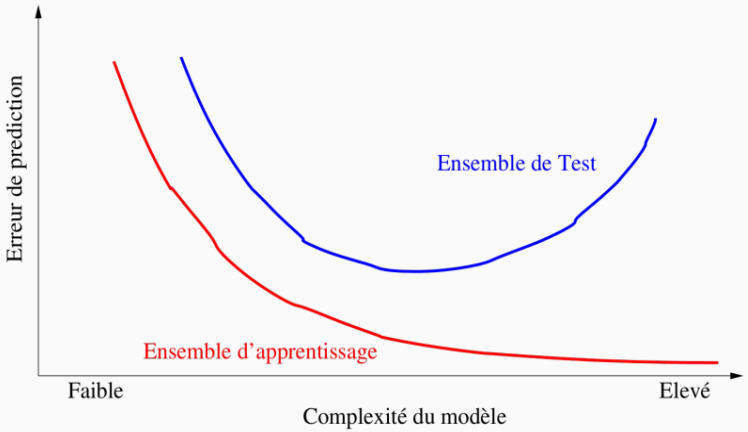
Lorsque le risque empirique est faible pour h^*, mais que l'erreur de généralisation est élevée, on parle de sur-apprentissage (overfitting).

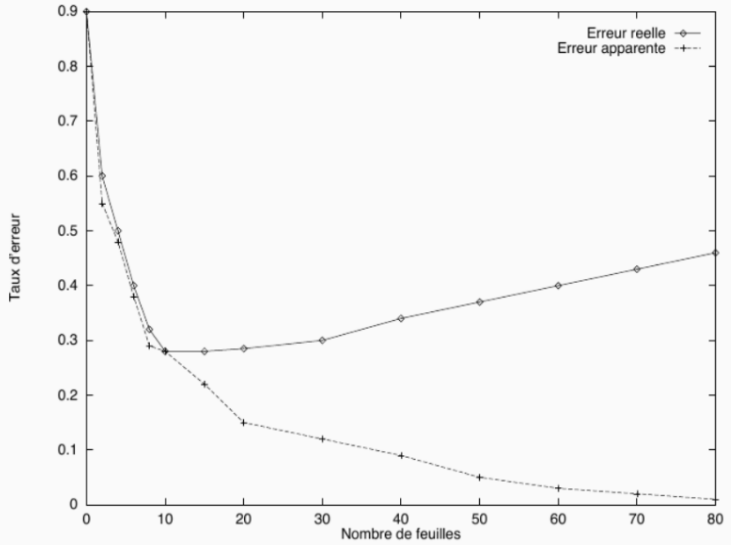
En revanche, le sous-apprentissage (underfitting) se produit lorsque le modèle est trop simple pour capturer la structure sous-jacente des données, même sur l'ensemble d'entraînement. Cela se traduit par un risque empirique plus élevé, car le modèle ne parvient pas à bien ajuster les données d'entraînement.

En résumé :

- Sur-apprentissage : Le modèle est trop complexe, s'ajuste trop précisément aux données d'entraînement, et ne généralise pas bien.
- Sous-apprentissage : Le modèle est trop simple, ne parvient pas à capturer la structure des données, et présente un risque empirique élevé.

Sur-apprentissage et Sous-apprentissage





Exemple de modèle linéaire : Prédiction des notes des élèves

Considérons un modèle linéaire pour prédire la note d'un élève (y) en fonction du nombre d'heures qu'il a étudié (x) :

$$y = \omega_0 + \omega_1 \cdot x + \epsilon$$

où :

- y est la note de l'élève,
- x est le nombre d'heures d'étude,
- ω_0 est l'intercept, la note attendue lorsque l'élève n'a pas étudié ($x = 0$), il est souvent associé au biais (bias), c'est la composante constante du modèle.
- ω_1 est la pente, représentant le changement attendu dans la note pour chaque heure d'étude supplémentaire,
- ϵ est le terme d'erreur résiduel qui capture toutes les influences non prises en compte dans le modèle.

Ce modèle permet d'estimer la note d'un élève en fonction du temps d'étude, en supposant une relation linéaire entre ces deux variables.

Erreurs dans le Contexte de Modélisation

Erreur Résiduelle :

- L'erreur résiduelle c'est la différence entre la valeur prédite et la valeur réelle dans les train data.
- Chaque point de données a une erreur résiduelle associée.

Objectif :

- L'objectif est de minimiser ces erreurs résiduelles, qui servent souvent à évaluer la qualité de l'ajustement du modèle aux données d'entraînement.

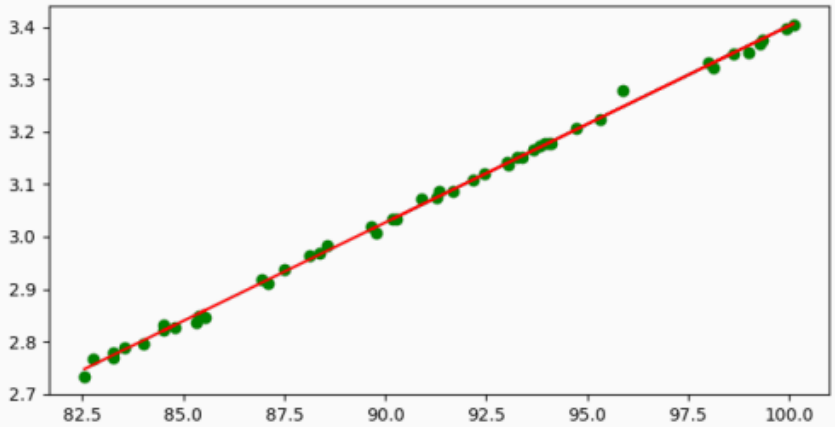
Récapitulatif :

- L'erreur réelle mesure la performance réelle du modèle sur de nouvelles données.
- L'erreur empirique évalue la performance sur l'ensemble d'entraînement.
- L'erreur résiduelle quantifie la différence entre les prédictions du modèle et les valeurs réelles dans l'ensemble d'entraînement.

Tracé des Points et Estimation de la Droite

Voici le tracé des points représentant les mesures, ainsi qu'une estimation de la droite :

$$y_i = \omega_0 + \omega_1 X_i$$



Les paramètres (ω_0, ω_1) sont estimés en minimisant l'erreur quadratique (moindres carrés).

Modèle de Régression Linéaire

La droite, déterminée par les paramètres estimés, agit comme un modèle de régression linéaire.

- Elle cherche à expliquer au mieux une variable cible Y en fonction de plusieurs autres variables $\{X^{(j)}\}_{j=1..d}$ (variables explicatives)
- Cette droite aide à séparer et quantifier les liens déterministes ainsi que les parties aléatoires (ϵ)
- On peut supposer, par exemple, que les ϵ suivent une distribution gaussienne centrée, avec l'estimation de l'écart-type σ
- Ce modèle offre la possibilité de prédire n'importe quelle valeur de la variable cible, comme une note, en fonction du nombre d'heures, par exemple.

Régression Linéaire Simple vs. Multiple

Si on a plusieurs variables explicatives (caractéristiques), on utilise la régression linéaire multiple.

- Avec une seule variable explicative, c'est une régression linéaire simple (comme dans l'exemple précédent).
- Le modèle est une ligne droite simple.
- Avec deux variables ou plus, c'est une régression linéaire multiple.
- La relation entre la réponse et les variables explicatives prend la forme :

$$y_i = b + w_1 x_i^{(1)} + w_2 x_i^{(2)} + \ldots + w_d x_i^{(d)} + u_i$$

Objectifs :

- Apprendre les paramètres w_j à partir des exemples de données.
- Déterminer les variables explicatives pertinentes : est-ce que la variable $X^{(j)}$ a une influence sur Y (c'est-à-dire, est-ce que $w_j \neq 0$?).
- Estimer l'erreur de prédiction du modèle.

Régression Linéaire Multiple

Donc si plusieurs variables explicatives sont présentes, on parle de régression linéaire multiple.

- Le modèle $h : \mathbb{R}^d \to \mathbb{R}$ à déterminer est de la forme :

$$h(x) = \sum_{i=1}^{d} w_i x^{(i)} + b = x^T w + b = [x^T 1]\alpha$$

Avec :

- $w \in \mathbb{R}^d$, un vecteur qui définit un hyperplan.
- $b \in \mathbb{R}$, un biais qui déplace la fonction perpendiculairement à l'hyperplan w.
- $\alpha = [w\ b] \in \mathbb{R}^{d+1}$.
-

$$[x^T 1]\alpha$$

En pratique, on cherche à déterminer (w, b) à partir de l'ensemble d'apprentissage D.

Représentation matricielle des données - Matrice augmentée

$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(d)} & 1 \\ x_i^{(1)} & x_i^{(2)} & \dots & x_i^{(d)} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(d)} & 1 \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Où :

- $x_i \in \mathbb{R}^d$ sont les observations (instances) pour $i = 1, \dots, n$.
- $y_i \in \mathbb{R}$ sont les valeurs observées à prédire (réponses) pour $i = 1, \dots, n$.
- $X \in \mathbb{R}^{n \times (d+1)}$ telle que $x = [x_1, x_2, \dots, x_n, e]$ avec $e \in \mathbb{R}^d$ et $e_i = 1, \forall i$.
- $y \in \mathbb{R}^n$ telle que $y = [y_1, y_2, \dots, y_n]^T$.

3.2 Méthode des moindres carrées ordinaire

Minimisation de l'Erreur de Prédiction

Pour déterminer $h(\cdot)$, nous cherchons à minimiser l'erreur de prédiction.

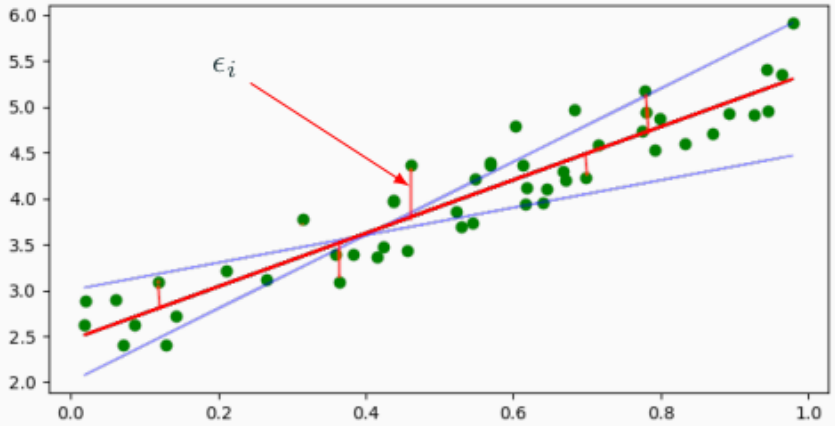
- Nous estimons $h(x) = x^T w + b$, nécessitant l'estimation des paramètres (w, b).
- La minimisation de l'erreur de prédiction sur les exemples d'apprentissage, également appelée résidu, est notre objectif.
- Les résidus, notés ϵ_i, sont définis comme
  $\epsilon_i = y_i - h(x_i)$
  $\epsilon_i = y_i - x_i^T w - b$,
  et peuvent être représentés sous forme matricielle : $\epsilon \in \mathbb{R}^n$,
  où $\epsilon = y - X\alpha$.

Interprétation Géométrique

On peut voir ce problème comme la recherche de l'hyperplan

$$y = x^T w + b$$

qui s'ajuste "au mieux" (selon la méthode des moindres carrés) parmi les observations
(x_i, y_i) pour $i = 1, \ldots, n$.



Minimisation des Résidus - Moindres Carrés

Les moindres carrés visent à minimiser la somme des carrés des résidus :

$$\min_h \sum_{i=1}^{n} (y_i - h(x_i))^2$$

$$\min_{(w,b)} \sum_{i=1}^{n} (y_i - x_i^T w - b)^2$$

Cela peut être formulé comme :

$$\min_{\alpha} \|y - X\alpha\|_2^2$$

Où $\|\cdot\|_2$ est la norme euclidienne, et $\|\epsilon\|_2 = \sum_{i=1}^{n} \epsilon_i^2$.

3.3 Qualité du modèle

Compromis Biais-Variance

En apprentissage, on évalue un modèle par son compromis biais-variance.

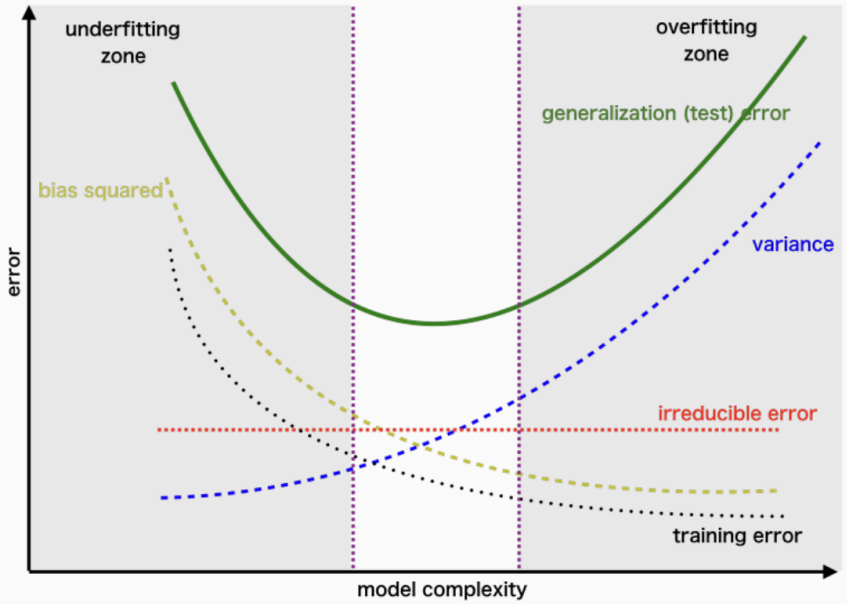
- **Biais** : Erreur due à des hypothèses incorrectes. Par exemple, si notre modèle est trop simple pour la vraie relation, il y a une erreur (biais non-nul).
- **Variance** : Erreur due à la sensibilité aux petits changements dans les données. Si le modèle réagit fortement à de petites variations, la variance est élevée (risque de sur-apprentissage).

Un bon modèle cherche un équilibre :

- Augmenter la complexité diminue le biais mais augmente la variance.
- Simplifier le modèle diminue la variance mais augmente le biais.

Underfitting, Overfitting, Compromis Biais-Variance

Underfitting : Modèle trop simple, biais élevé.
Overfitting : Modèle trop complexe, variance élevée.
Compromis biais-variance : Équilibre pour une bonne généralisation.



Estimation du Biais et de la Variance

Pour évaluer les performances d'un modèle, nous devons le tester sur des données distinctes de celles utilisées pour l'apprentissage.

- Divisons les données en deux parties : un ensemble d'apprentissage et un ensemble de test.

Les métriques clés (erreur, biais, variance) doivent être évaluées pour plusieurs modèles formés sur différents ensembles d'apprentissage.

- Répétons les phases d'apprentissage/test sur plusieurs découpages de données.

Ces métriques caractérisent la performance d'un algorithme d'apprentissage.

- Utilisons le même algorithme pour toutes les répétitions.

Dans ces estimations, nous ignorons σ^2 et remplaçons $h(x)$ par les y associées aux données supervisées.

Évaluation des Performances d'un Modèle de Régression

Lorsqu'on veut évaluer la qualité d'un modèle, deux mesures courantes sont utilisées sur des données de test.

Erreur Quadratique Moyenne (MSE):

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

- La MSE est nulle lorsque les prédictions sont parfaites.
- Cependant, elle n'est pas normalisée et dépend de la variance de y.

Évaluation des Performances d'un Modèle de Régression

Coefficient de Corrélation (r):

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sigma_y \sigma_{\hat{y}}}$$

- $$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

  est la moyenne de y et σ_y est sa variance.
- Le coefficient de corrélation est 1 lorsque les prédictions sont parfaites.
- C'est une mesure normalisée.

3.3 Méthode des moindres carrées régularisés - regression ridge

Problème des Moindres Carrés Ordinaires

- Problème lorsque $n < d + 1$: $X^T X$ non inversible.
- La solution n'est pas unique, problème mal posé.
- Solution : régularisation.

Idée de la Régularisation

- Ajout d'une contrainte sur les paramètres à estimer.
- Cette contrainte résout le problème de $X^T X$ non inversible.
- Objectif ML : pénaliser les modèles trop complexes pour éviter le sur-apprentissage.
- Contrôlé par un hyperparamètre de régularisation λ.

Régression Ridge (régularisation de Tikhonov)

- La régression Ridge vise à minimiser la norme de w tout en maintenant la solution du système linéaire stable, avec λ limite le sur-apprentissage.
- Objectif : $\min_{w,b} \frac{1}{2} \left(\sum_{i=1}^{n} \left(y_i - x_i^T w - b \right)^2 + \lambda \sum_{i=1}^{d} w_i^2 \right)$.
- La régularisation de Tikhonov est appliquée avec un terme de pénalité quadratique (l_2) sur les poids (w_i).
- Effet : Promouvoir des paramètres w de norme minimale, rendant le problème strictement convexe.
- L'hyperparamètre λ contrôle l'intensité de la régularisation.
- Pour $\lambda = 0$, on retrouve la régression des moindres carrés.
- La méthode résultante est appelée régression Ridge ou régression de crête.

Trouver la Valeur de λ

- λ contrôle la quantité de régularisation.
- Pour chaque valeur de λ on a une solution différente donc on veut trouver la valeur de λ qui minimise la performance (MSE), autrement, en choisissant différentes valeurs de λ, vous obtiendrez des modèles avec des niveaux de complexité différents. L'idée est de trouver la valeur de λ qui donne le meilleur compromis entre ajustement aux données d'entraînement et généralisation à de nouvelles données en donnant la plus faible erreur quadratique moyenne.
- Solution : mesurer les performances sur des données de test pour plusieurs valeurs de λ.
- Techniques pour fiabiliser : cross-validation, bootstrap, leave-one-out.

Régression LASSO et Elastic Net

- Régression LASSO utilise la norme l_1.
- Peut sélectionner les variables explicatives en autorisant certains w_i à être 0.
- Elastic Net combine Ridge et LASSO.
- Corrige les défauts de LASSO.

Régression LASSO, Ridge et Elastic Net

- Une autre méthode de régularisation populaire est la méthode LASSO.
- Norme l_p : $||x||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$.
- La Régression Ridge est basée sur la norme l_2 (norme euclidienne).
- La Régression LASSO (Least Absolute Shrinkage and Selection Operator) utilise la norme l_1.
- Objectif LASSO : $\mathbf{min}_{w,b} \frac{1}{2} \left(\sum_{i=1}^{n} \left(y_i - x_i^T w - b\right)^2 + \lambda \sum_{i=1}^{d} |w_i|\right)$.
- Elastic Net combine les deux méthodes : $\mathbf{min}_{w,b} \frac{1}{2} \left(\sum_{i=1}^{n} \left(y_i - x_i^T w - b\right)^2 + \lambda_1 \sum_{i=1}^{d} w_i^2 + \lambda_2 \sum_{i=1}^{d} |w_i|\right)$.
- Pour Elastic Net, avec $\lambda_1 = 0$ et $\lambda_2 > 0$, c'est équivalent à Ridge ; avec $\lambda_1 > 0$ et $\lambda_2 = 0$, c'est équivalent à LASSO ; avec $\lambda_1 = \lambda_2 = 0$, c'est équivalent à la Régression des Moindres Carrés (MCO).
- Elastic Net corrige un défaut de LASSO lorsque d est grand et n est petit, évitant la sélection de variables non pertinentes (au maximum n variables sélectionnées)

3.1 Regression logistique

Classification et Régression : Rappel

- Pour revisiter, souvenons-nous des deux grandes catégories de problèmes d'apprentissage supervisés :
  1. Classification : Y est un ensemble discret de c valeurs : $Y = \{\lambda_1, \lambda_2, \ldots, \lambda_c\}$ avec $c \geq 2$. Un modèle dans ce contexte est appelé un classifieur.
  2. Régression : Y est un ensemble continu de réels : $Y \subset \mathbb{R}$. Un modèle dans ce contexte est appelé un régresseur.
- Les méthodes précédentes se concentrent sur la problématique de la régression.
- Ici nous voulons souligner l'existence des méthodes de régression adaptées à des tâches de classification.

Classification Binaire : Adaptation pour la Régression

- Dans le contexte de la classification binaire, une adaptation directe est réalisable.
- Pour simplifier, concentrons-nous sur la classification binaire ($c = 2$).
- Définissons les classes comme suit : $\lambda_1 = 1$ et $\lambda_2 = -1$.
- Notre objectif est de trouver un modèle $h : \mathbb{R}^d \to \{-1, 1\}$ en utilisant $X \in \mathbb{R}^{n \times d}$.
- La classe prédite est déterminée par le signe de $h(\cdot)$.

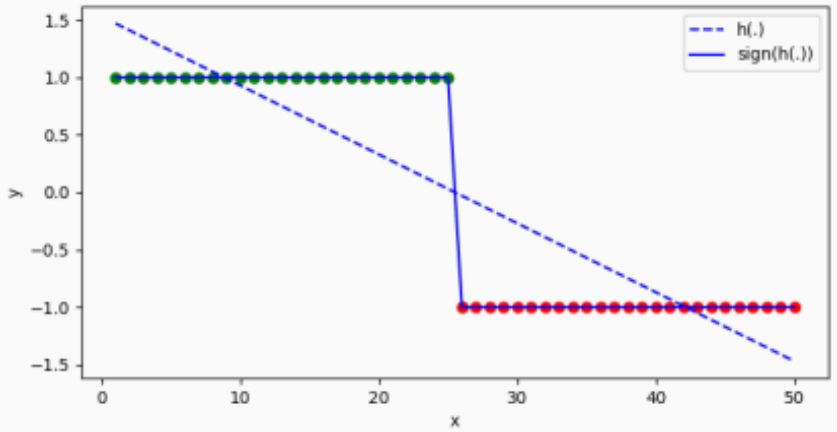


Figure 1: recherche du bon h qui sépare les classes 1 et -1

Défauts de l'Adaptation Directe

- Elle présente plusieurs inconvénients :
  1. Pas de généralisation naturelle pour plus de 2 classes (cas fréquent dans le monde réel).
  2. Les résidus n'ont pas de signification claire, comme illustré dans l'exemple précédent.
  3. Sensibilité élevée aux données d'apprentissage, avec des résultats parfois peu pertinents.
- Exemple de l'image de la slide 34 : Bien que le problème ne soit pas plus complexe, les résultats ne sont pas satisfaisants.

Prédiction de Probabilité pour la Classification Binaire

- Une approche alternative consiste à anticiper la probabilité d'appartenance aux classes.
- Limitons-nous toujours à la classification binaire.
- Considérons cette fois-ci $\lambda_1 = 0$ et $\lambda_2 = 1$.
- L'objectif n'est plus de prédire y, mais la probabilité d'appartenance à la classe λ_1, notée $p(\lambda_1|x)$.
- Utilisons la régression linéaire pour estimer le modèle : $\pi(x) = x^T w + b$, où $\pi(x)$ représente la probabilité.

Problème de Modélisation : Régression Linéaire

- Problème : La prédiction $\pi(x)$ appartient à l'interval $[0, 1]$ (ensemble continue ou R)
- Et dès lors, le modèle linéaire n'est pas adapté pour modéliser une probabilité

Un modèle de régression logistique

Modèle de Régression Logistique

- Utilisation de la fonction logit :
  $\text{logit}(p(1|x)) = w_1 x^{(1)} + w_2 x^{(2)} + \ldots + w_d x^{(d)} + b$

Modèle de Régression Logistique (Forme Matricielle)

- Modèle de régression logistique : $h(x) = \sigma(w^T x + b)$ notre fonction de prédiction, où $\sigma(z) = \frac{1}{1+e^{-z}}$ (sigmoïd)
- La fonction $h(x)$ prédit $p(1|x)$ sous forme de probabilité.

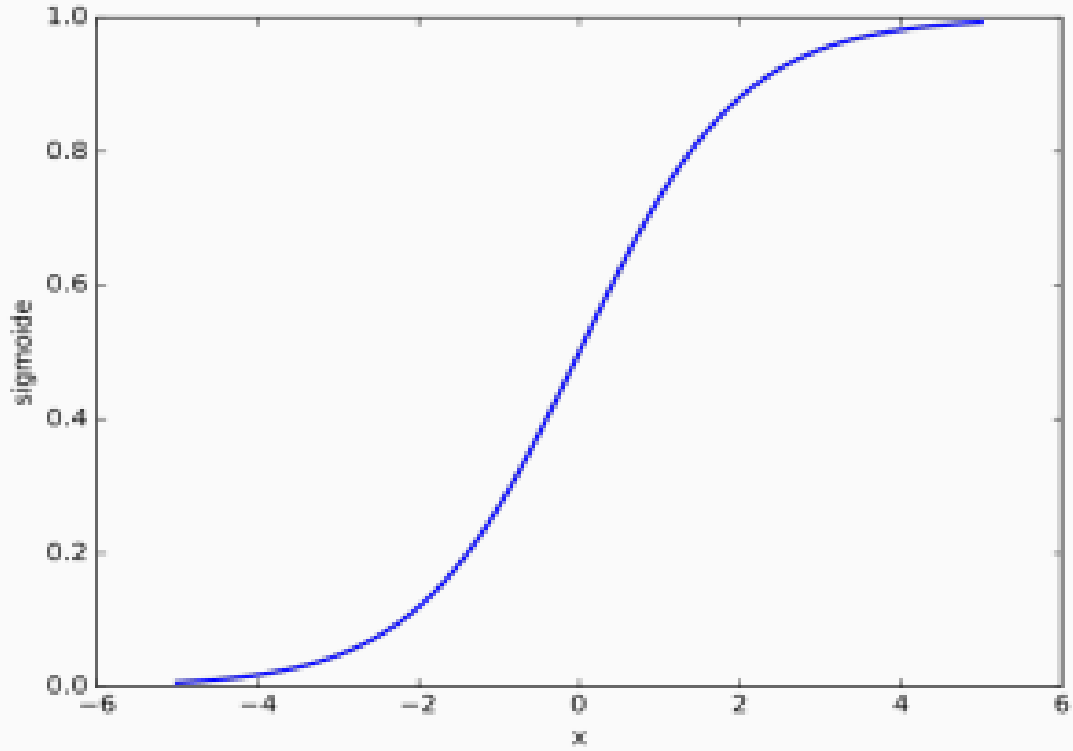


Figure 2: Elle remplace la foction linéaire

Modèle de Régression Logistique (Forme Matricielle)

- Sous forme matricielle, le modèle de régression logistique est donné par :

$$h(x) = \pi(x) = \sigma(w^T x + b) = \frac{1}{1 + \exp\left(-w^T x - b\right)}$$

- La fonction $h(x)$ est une prédiction de $p(\lambda_1|x)$ plutôt que y.
- Pour la classification binaire : $p(\lambda_1|x) = 1 - p(\lambda_2|x)$.
- Une autre expression équivalente : $p(\lambda_2|x) = 1 - \frac{1}{1+\exp\left(-w^T x - b\right)}$.
- Cette équation peut aussi être écrite comme $p(\lambda_2|x) = \frac{1}{1+\exp\left(w^T x + b\right)}$.
- Toutes ces expressions montrent que $p(\lambda_2|x) = \sigma(-(w^T x + b))$.

Apprentissage du modèle de régression logistique

Maximum de Vraisemblance - Objectif

L'objectif est d'**estimer les paramètres du modèle de régression logistique en maximisant la vraisemblance** (que nous n'allons aps trop détaillé ici).

- Utilisation de la vraisemblance conditionnelle des données d'apprentissage pour estimer les paramètres.
- **On veut maximiser la probabilité conditionnelle des classes y_i sachant les observations x_i.**

Estimation des paramètres θ par maximisation de la probabilité conditionnelle des classes y_i sachant x_i.

Maximum de Vraisemblance - Log-Vraisemblance et Coût

- La log-vraisemblance est utilisée comme fonction objectif à minimiser.
- La fonction de coût correspondante pour la régression logistique est définie par la négation de la log-vraisemblance.
- Elle prend en compte la probabilité d'obtenir les classes réelles y_i avec le modèle paramétré par θ.

Fonction de coût :

$$\mathrm{LL}(\theta) = -\sum_{i=1}^{n} \left(y_i \log(p(1|x_i;\theta)) + (1-y_i) \log(p(0|x_i;\theta)) \right)$$

avec $\quad \mathrm{p}(\lambda_1|x) = \frac{1}{1+e^{-z}}, \quad p(\lambda_2|x) = \frac{1}{1+e^{-z}}, \quad$ et $\quad z = w^T x + b$

Maximum de Vraisemblance - Formulation

Partant de cette formulation de la fonction cout,on la développe pour obtenir la loss de la logstique :

$$
\begin{aligned}
LL(\theta) &= -\sum_{i=1}^{n}\left[y_i \log\left(\frac{1}{1+e^{-z_i}}\right) + (1-y_i)\log\left(1-\frac{1}{1+e^{-z_i}}\right)\right] \\
&= -\sum_{i=1}^{n}\left[y_i z_i - \log\left(1+e^{z_i}\right)\right] \\
&= -\sum_{i=1}^{n}[y_i z_i - \log(1+e^{z_i})] \\
&= -\sum_{i=1}^{n} y_i z_i + \sum_{i=1}^{n}\log(1+e^{z_i}) \\
&= -\sum_{i=1}^{n} y_i (w^T x_i + b) + \sum_{i=1}^{n}\log(1+e^{w^T x_i + b}) \\
&= -\sum_{i=1}^{n} y_i (w_1 x_{i1} + w_2 x_{i2} + \ldots + w_d x_{id} + b) + \sum_{i=1}^{n}\log(1+e^{w^T x_i + b})
\end{aligned}
$$

Maximum de Vraisemblance et Descente de Gradient

Méthodes itératives pour l'estimation des paramètres :

- Il existe plusieurs méthodes itératives pour résoudre le problème d'optimisation associé à la maximisation de la vraisemblance.
- Nous n'énumérons pas toutes les méthodes ici, mais nous illustrons avec un exemple : la méthode de descente de gradient.
- La descente de gradient implique le calcul du gradient de la log-vraisemblance par rapport aux paramètres du modèle.
- Une fois le gradient calculé, la mise à jour des paramètres est effectuée pour minimiser la log-vraisemblance.
- Cela se fait itérativement, en ajustant les paramètres dans la direction opposée au gradient.

Maximum de Vraisemblance et Descente de Gradient

Formule de la descente de gradient :

$$w_k \leftarrow w_k - \gamma \sum_{i=1}^{n} x_i \left(y_i - \frac{e^{-z_i}}{1 + e^{-z_i}} \right)$$

Ici, γ est le taux d'apprentissage, w_k est le k-ème paramètre du modèle, et $z_i = w^T x_i$.

Régression Logistique en Classification Multiclasse

Les termes

Extension au cas multiclasse :

- Le modèle de régression logistique précédent est conçu pour la classification binaire (c = 2).
- Pour le cas multiclasse, on utilise la régression logistique multinomiale, également appelée régression softmax.
- Chaque classe λk est modélisée individuellement avec la stratégie one-vs-all.
- La fonction softmax est utilisée pour calculer la probabilité d'appartenance à chaque classe en considérant tous les modèles de classe.
- Cette approche permet de gérer efficacement un nombre arbitraire de classes.
- Pour plus de détails, référez-vous aux travaux pratiques (TP) ou cours associés.

Regression Logistique en Classification Multiclasse

Pour la regression logistique **binaire** à une classe nous avons vu la fonction **sigmoïd**, pour la regression **multiclasse**, la probabilité ou la fonction logit s'appelle **softmax** .

Formule de la fonction softmax :

$$p(y = \lambda_k | x) = \frac{\exp(x^T w_k)}{\sum_{j=1}^{C} \exp(x^T w_j)}$$

Ici, C est le nombre de classes, w_k est le vecteur paramètre du modèle pour la classe λ_k, et x est le vecteur d'entrée.

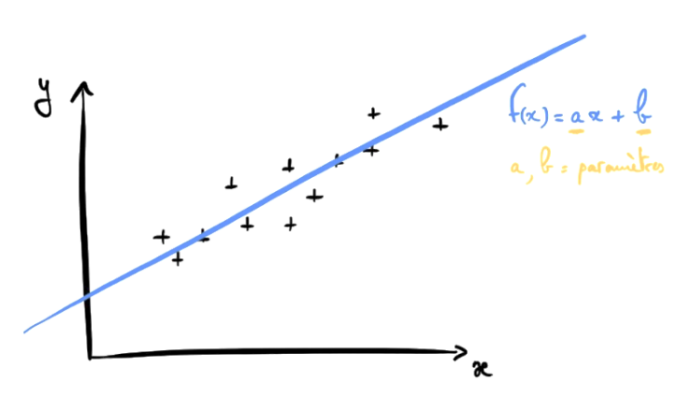
Remarque : Chaque classe a son propre modèle, et le choix de la classe prédite est basé sur la probabilité softmax.

Comprendre la descente de gradient simplement

Soit un modèle linéaire

on a vu à quoi ressemble un modèle linéaire : **f(x)=ax+b** .

Sauf que nous ne connaissons pas les valeurs de a et b, qui seront décidés par le modèle de ML de sorte à tracer un modèle qui s'insère bien dans notre nuage de point :



1. la droite : c'est le modèle trouvé grace aux valeur de paramètres a et b
2. les croix sont les valeurs de Test
3. l'axe x : ce sont les valeurs X_i des données de test
4. l'axe y : ce sont les labels

Cette partie s'inspire des notions partagées par G. Saint-Cirgue .

La fonction coût de ce modèle linéaire

Dans le cas de la regression linéaire on utilise la || . || (norme euclidienne) pour mesurer l'erreur entre y et $f(x_{(i)})$.

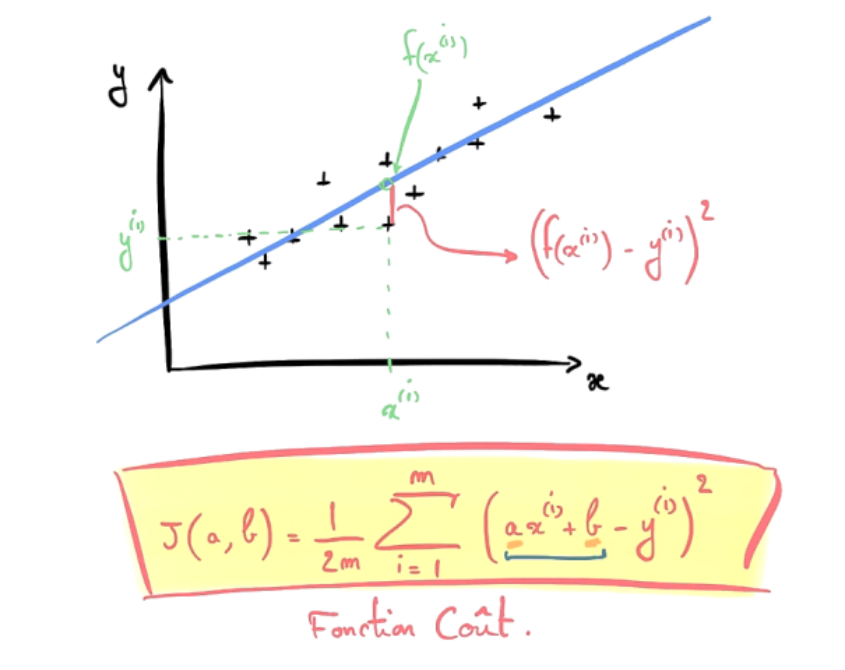
C'est à dire que **l'erreur de prédiction** (résiduelle) se trouve de cette façon :

$\textbf{erreur}_{(i)} = (f(x_{(i)}) - y_{(i)})$.
avec f(x) qui correspond aux predictions .

Ici, l'erreur désigne une composante de la fonction de coût .

Supposons par exemple que à i = 10, $X_{(10)} = 80m^{(2)}$
comme surface de l'appartement, avec un prix $y_{(10)} = 100,000EUR$.
Et malheureusement $f(x_{(10)}) = 100,002EUR$.
Cela signifie qu'il y'a une erreur de prédiciton de
$(100{,}002\text{-}100{,}000)^2 = (2)^2 = 4EUR$.

La fonction coût de ce modèle linéaire



1. La droite : c'est le modèle trouvé grace aux valeur a et b .
2. $X_{(i)} et Y_{(i)}$
   : ce sont les valeurs X et Y de la 10 eme ligne de notre dataset .
3. La droite verticale en rouge : c'est l'erreur, la distance entre la valeur réelle et la prédiction .
4. La foncion cout : c'est la moyenne des toutes les i ème erreurs des données de tests (l'erreur quadratique moyenne).

La Descente de gradient

La descente de gradient, c'est la recherche (par la machine) des paramètres optimaux qui minimisent la fonction coût en trouvant le minimum de cette fonction,

autrement,

on cherchera les paramètres qui nous donnent le meilleur modèle qui nous permettra d'avoir produira moins d'erreurs résiduelles.

Scénario de la Descente de gradient

Figure 3: le scénario de la montagne et du refuge

le principe de l'algorithme de descente de gradient peut s'expliquer en 4 étapes

1. Etape 0 (initialisation) : on commence avec des valeurs a et b de départ.
2. Etape 1 (la descente ou la pente) : c'est la dérivée de j(a,b)
3. Etape 2 (le pas de descente) : c'est l'orientation et la distance qu'on prend
4. Etape 3 (la boucle) : on continue jusqu'à atteindre le minimum .

La Descente de gradient

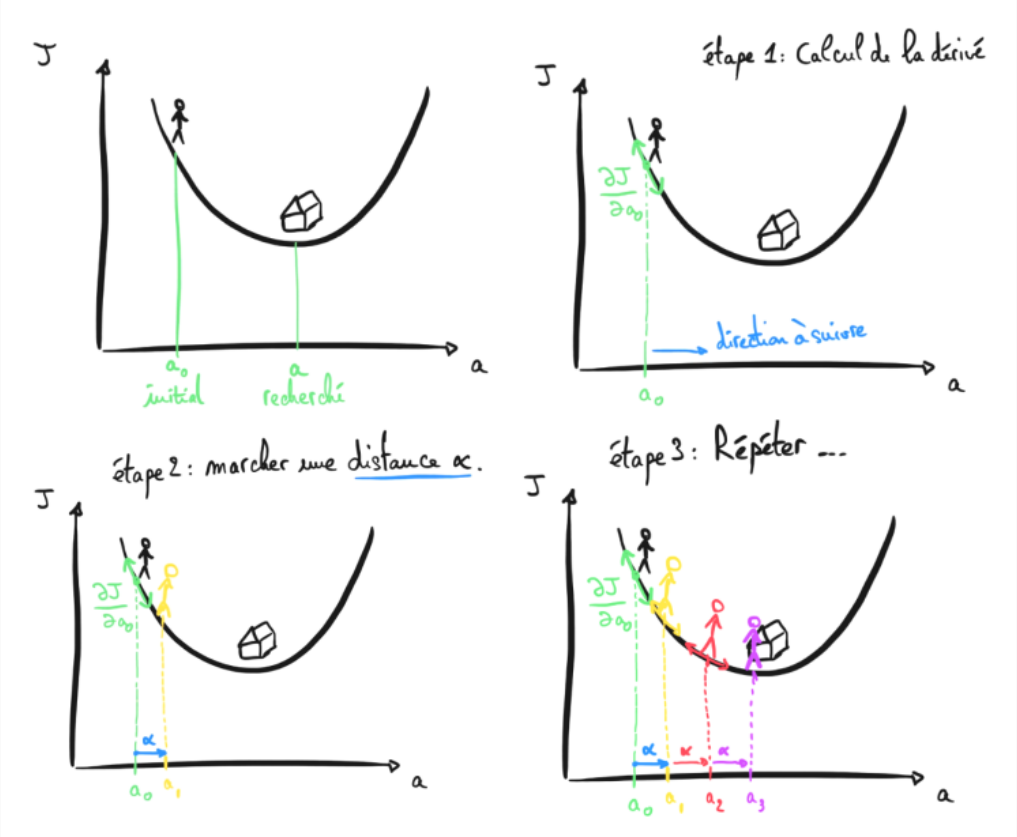


Figure 4: les 4 étapes

En ordonnée on a **la fonction de cout** et **en abscisse** on a **le pas** (alpha) de decente traduit .

Les valeurs A et B

A chaque étape 2 : les valeurs des paramètres a et b s'actualisent (c'est ça le but : trouver les meilleurs a et b on s'attend donc à trouver les meilleurs a et b au minimum, car els meilleurs a et b minimisent J) .

Répéter en boucle (les dérivées partielles ou gradient de J):

$$\mathsf{a} = \mathsf{a} - \alpha \frac{\partial \mathcal{J}(\mathsf{a}, \mathsf{b})}{\partial \mathsf{a}}$$

$$\mathsf{b} = \mathsf{b} - \alpha \frac{\partial \mathcal{J}(\mathsf{a}, \mathsf{b})}{\partial \mathsf{b}}$$

cela signifie que à chaque itération, le nouveau paramètre devient, l'ancienne valeur moins la pente multipliée par la distance à parcourir (pas) .
le paramètre alpha désigne le learning rate ou taux d'apprentissage .

Le pas alpha : le learning rate

Le ”learning rate” est un hyperparamètre de valeur fixe qui contrôle la taille des pas que l'algorithme de descente de gradient prend lors de la mise à jour des paramètres du modèle .

Il faut bien choisir ces valeurs car l'échelle de grandeur a un grand impacte sur la recherche du minimum (trop petit, trop de temps).

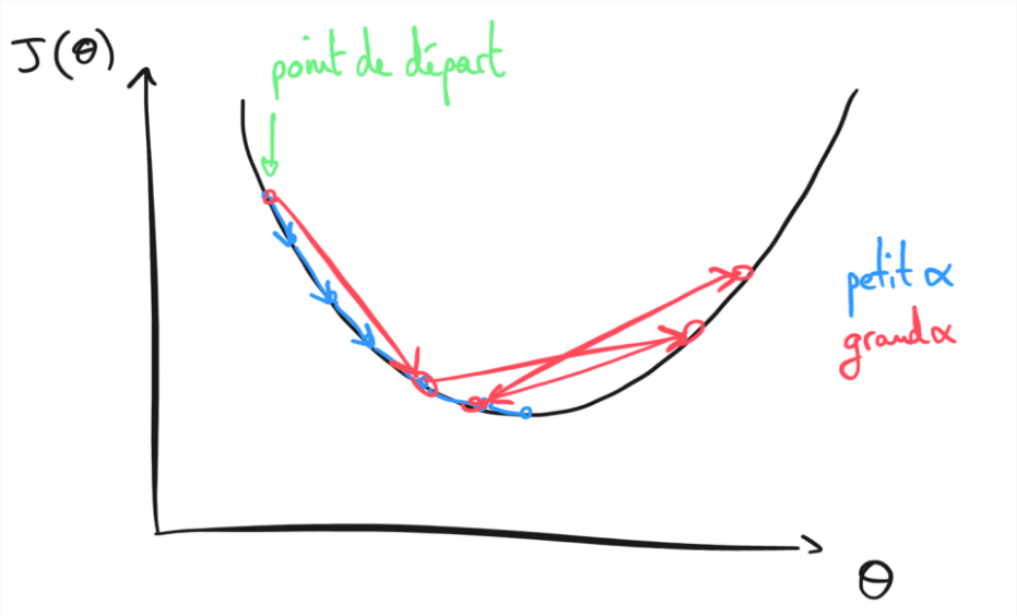


Figure 5: Choix du learning rate