Introduction au machine learning

Redha Moulla

Plan du cours

- Machine learning : généralités
- Machine learning : apprentissage supervisé
- Évaluation et sélection des modèles
- Machine learning : apprentissage non supervisé
- Prétraitement des données
- Approche bayésienne
- Introduction au deep learning

Machine learning

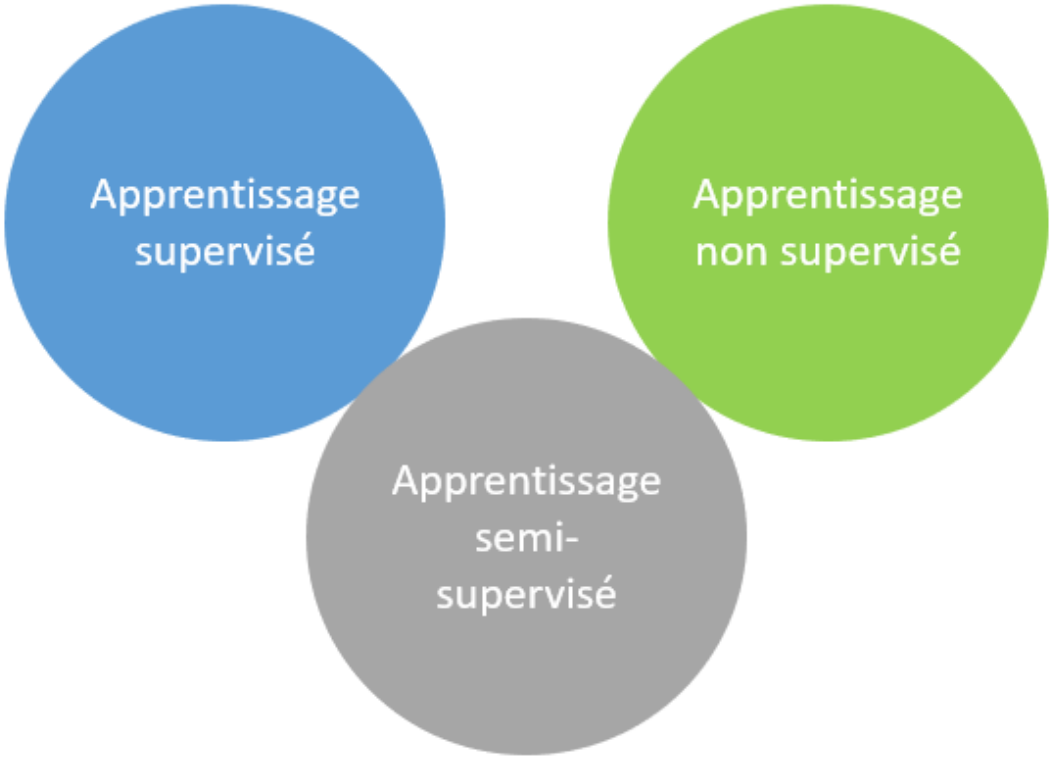
Définition de l'apprentissage automatique

L'apprentissage automatique est une branche de l'intelligence artificielle qui consiste à doter les machines de la capacité d'apprendre à partir de données sans que celles-ci ne soient explicitement programmées pour exécuter des tâches spécifiques.

Le machine learning englobe plusieurs types d'apprentissage :

- **Supervisé** : Les algorithmes apprennent à partir de données étiquetées pour faire des prédictions ou classifications.
- **Non supervisé** : L'apprentissage est effectué sur des données non étiquetées pour trouver des structures cachées.
- **Semi-supervisé** : Combine des éléments des deux premiers types en utilisant une petite quantité de données étiquetées et une grande quantité de données non étiquetées.
- **Par renforcement** : Les modèles apprennent à prendre des décisions en maximisant une récompense à travers des interactions.

Typologies d'apprentissage automatique



L'apprentissage supervisé

L'apprentissage supervisé consiste à apprendre un modèle qui associe une étiquette (*label*) à un ensemble de caractéristiques (*features*).

- **Entrées** : un jeu de données *annotées* pour entraîner le modèle.
  - Exemple : des textes avec les sentiments associés, positifs ou négatifs.
- **Sortie** : une étiquette ou une valeur pour une observation non vue.

L'apprentissage supervisé se décline lui-même en deux grandes familles :

- **La classification** : prédire une catégorie ou une classe.
  - Exemple : prédire l'étiquette d'une image (chat, chien, etc.), le sentiment associé à un texte, le centre d'intérêt d'un client à partir de ses commentaires, etc.
- **La régression** : prédire une valeur continue (un nombre réel typiquement).
  - Exemple : prédire le prix d'un appartement, la lifetime value d'un client, etc.

L'apprentissage non supervisé

L'apprentissage non supervisé se réfère à l'utilisation de modèles d'apprentissage automatique pour identifier des patterns et des structures dans des données qui ne sont pas étiquetées.

Principales typologies de l'apprentissage non supervisé :

- **Partitionnement** : regroupement d'observations similaires. Exemple : segmentation de marché.
- **Réduction de dimensionnalité** : Techniques pour réduire la dimension des données tout en préservant leur structure. Exemple : visualisation de données en grande dimension.
- **Détection d'anomalies** : Détecter des observations dont les caractéristiques sont inhabituelles par rapport à la majorité.

L'apprentissage semi-supervisé

L'apprentissage semi-supervisé combine des éléments des approches supervisées et non supervisées. Il utilise un petit ensemble de données étiquetées et un plus grand ensemble de données non étiquetées pour former des modèles.

Cette méthode est particulièrement utile quand :

- Les données étiquetées nécessitent des ressources coûteuses pour les obtenir, mais les données non étiquetées sont abondantes.
- L'ajout d'un peu d'information étiquetée peut améliorer significativement la performance de modèles entraînés avec des données non étiquetées.

Les applications typiques incluent :

- Classification d'images ou de textes lorsque les annotations sont rares.
- Analyse de sentiment avec un petit corpus annoté et un grand corpus non annoté.

L'apprentissage par renforcement

L'apprentissage par renforcement est une approche de l'apprentissage automatique où un agent apprend à prendre des décisions en interagissant avec un environnement. L'objectif est de maximiser une récompense cumulative.

Principes clés de l'apprentissage par renforcement :

- **Exploration et exploitation** : l'agent doit explorer l'environnement tout en exploitant ses connaissances actuelles.
- **Politique** : Une stratégie qui guide l'agent à choisir une action à partir d'un état donné.
- **Récompense** : Un signal immédiat reçu après chaque action, qui aide à évaluer la performance.
- **Valeur** : espérance du retour futur à partir d'un état, éventuellement conditionnée par une action.

L'apprentissage par renforcement est largement utilisé dans les domaines tels que la robotique, les jeux vidéos, les enchères, la recommandation, etc.

Apprentissage supervisé

Principes de l'apprentissage supervisé

La construction d'un modèle en apprentissage supervisé repose sur trois piliers essentiels qui guident le processus d'apprentissage et déterminent sa réussite :

- **Données** : L'ensemble d'exemples étiquetés utilisés pour l'entraînement du modèle. La qualité, la quantité et la représentativité de ces données sont cruciales pour la capacité du modèle à apprendre et à généraliser à de nouvelles observations.
- **Fonction objective** : Aussi connue sous le nom de fonction de perte ou de coût, elle quantifie l'erreur entre les prédictions du modèle et les valeurs réelles. L'objectif de l'apprentissage est de minimiser cette fonction, guidant ainsi le modèle vers une meilleure performance.
- **Hypothèses (biais inductif)** : Les hypothèses préalables sur la forme du modèle et les relations dans les données. Ces hypothèses concernent le choix de l'algorithme d'apprentissage, et toute préconception sur la distribution des données. Ces hypothèses dirigent l'espace de recherche des solutions possibles et influencent directement la capacité du modèle à apprendre et à généraliser.

Formalisation du problème de l'apprentissage supervisé

Soient les observations $x_1, \ldots, x_n \in \mathcal{X}$ et les réponses correspondantes $y_1, \ldots, y_n \in \mathcal{Y}$.
L'apprentissage supervisé consiste à construire une fonction $f : \mathcal{X} \to \mathcal{Y}$ qui généralise à de nouvelles observations, à partir de l'échantillon $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$.

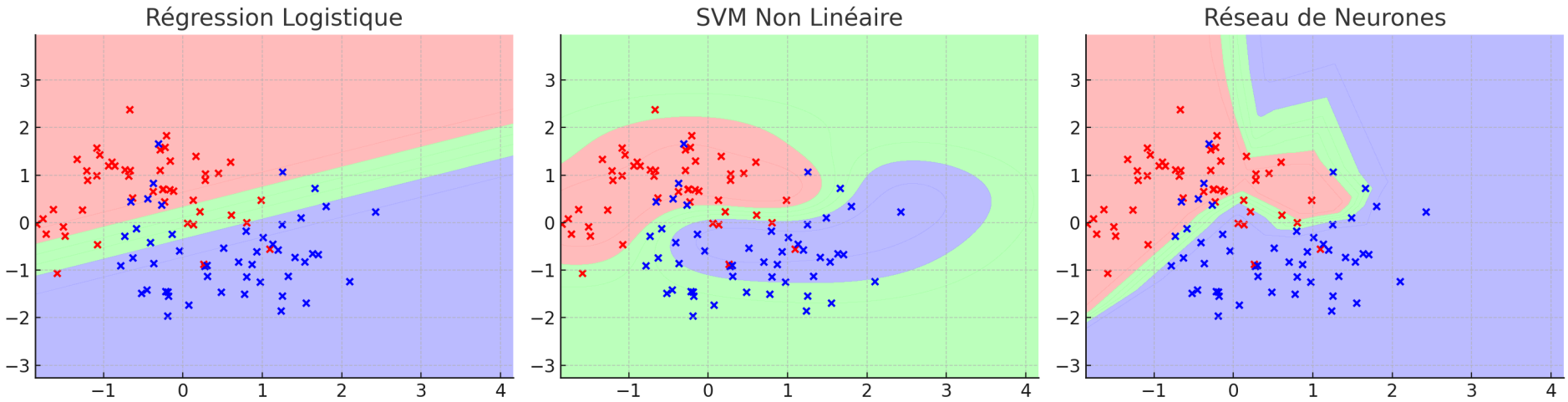
On parle de :

- Régression quand $\mathcal{Y} = \mathbb{R}$
- Classification quand $\mathcal{Y} = \{1, 2, \ldots, C\}$, où $C \geq 2$.

Espace des solutions

Comment choisir la fonction f dans l'ensemble des solutions $\mathcal{F}$?

- A priori sur la classe du modèle (biais inductif)
- Minimisation du risque empirique



Biais inductif

Le biais inductif désigne l'ensemble des hypothèses qui permettent à un algorithme de généraliser au-delà des observations d'entraînement. Il intervient notamment dans le choix de la classe de modèles, de la régularisation et de la procédure d'apprentissage.

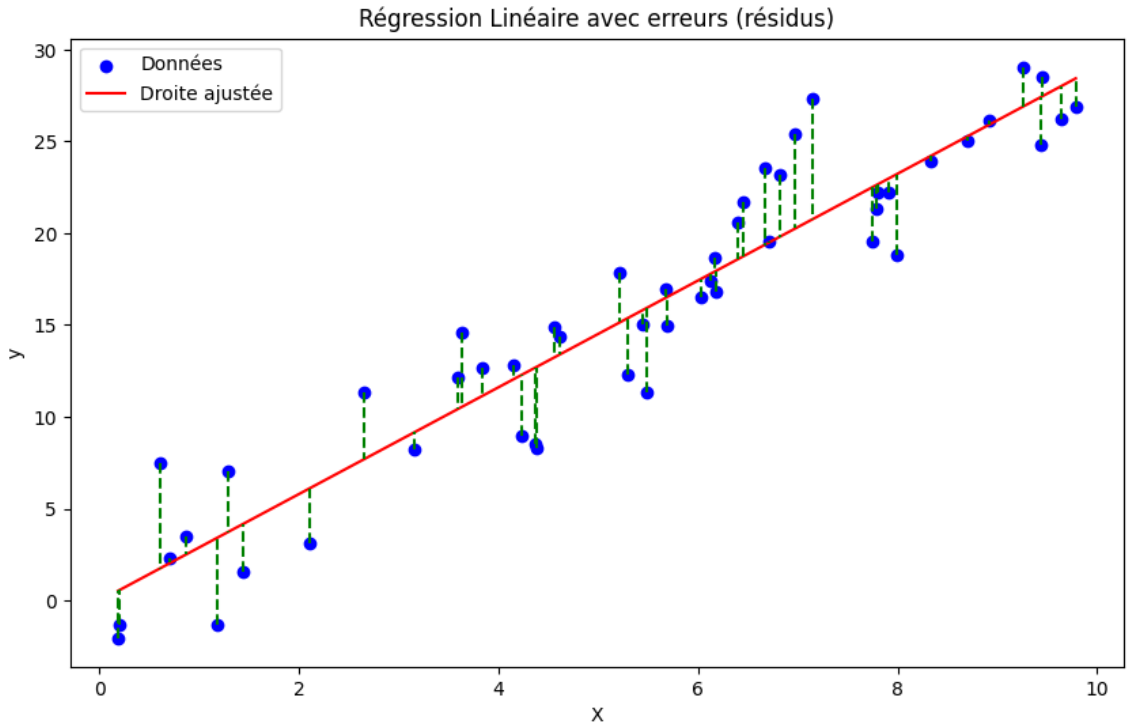
Exemple de biais inductif :

- Hypothèse de linéarité.
- Hypothèse de proches voisins.
- Hypothèse de maximum de marge.
- Hypothèse d'équivariance par translation.

Minimisation du risque empirique 1/2

Une fois la classe $\mathcal{F}$ choisie, on ajuste un modèle en minimisant une fonction de perte L sur les données d'entraînement.

$$\widehat{R}_n(h) = \frac{1}{n} \sum_{i=1}^{n} L\big(h(x_i), y_i\big)$$

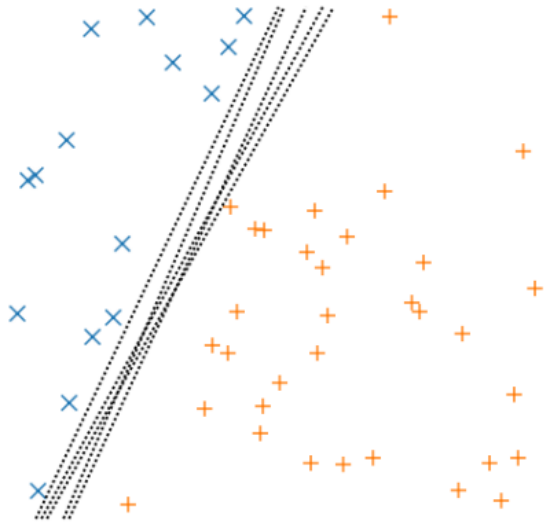


Minimisation du risque empirique 2/2

Un estimateur empirique vérifie :

$$\widehat{f} \in \arg\min_{h \in \mathcal{F}} \widehat{R}_n(h).$$

Une faible erreur d'entraînement ne garantit pas une faible erreur sur de nouvelles données. La généralisation dépend de la taille et de la représentativité de l'échantillon, de la complexité de $\mathcal{F}$ et de la régularisation.



Risque et généralisation

Le risque population mesure l'erreur moyenne sur de nouvelles observations :

$$R(h) = \mathbb{E}_{(X,Y)\sim P}\left[L(h(X), Y)\right].$$

Le risque empirique $\widehat{R}_n(h)$ n'en est qu'une estimation. L'écart $R(h) - \widehat{R}_n(h)$ dépend notamment :

- de la taille et de la représentativité de l'échantillon ;
- de la complexité de la classe de modèles ;
- du nombre de choix effectués à partir des données ;
- d'un éventuel changement de distribution entre entraînement et utilisation.

Fonctions de coût

Le choix de la fonction de coût dépend de la nature du problème et des données. On distingue deux types de fonctions de coût.

Régression

- L'erreur quadratique : $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$
- L'erreur absolue : $L(\hat{y}, y) = |\hat{y} - y|$

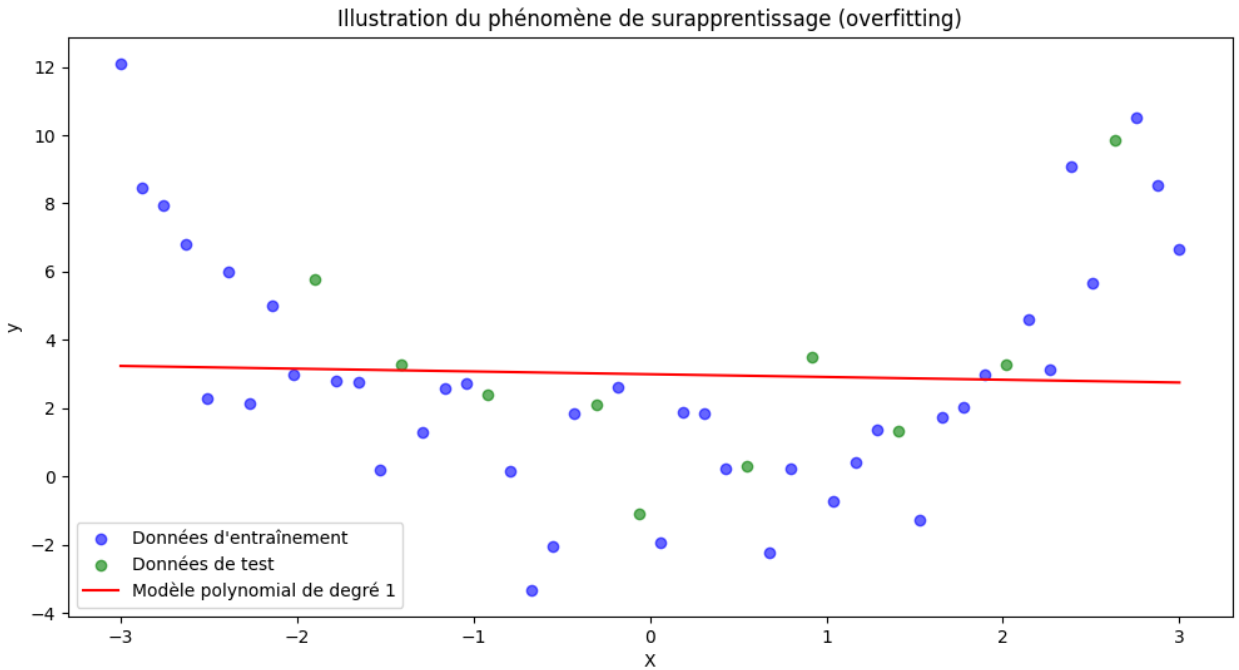
Classification

- Entropie croisée binaire : $L(p, y) = -y \log p - (1 - y) \log(1 - p)$, avec $y \in \{0, 1\}$.
- Perte hinge : $L(s, y) = \max(0, 1 - ys)$, avec $y \in \{-1, +1\}$ et un score $s \in \mathbb{R}$.

Sous-apprentissage

Definition

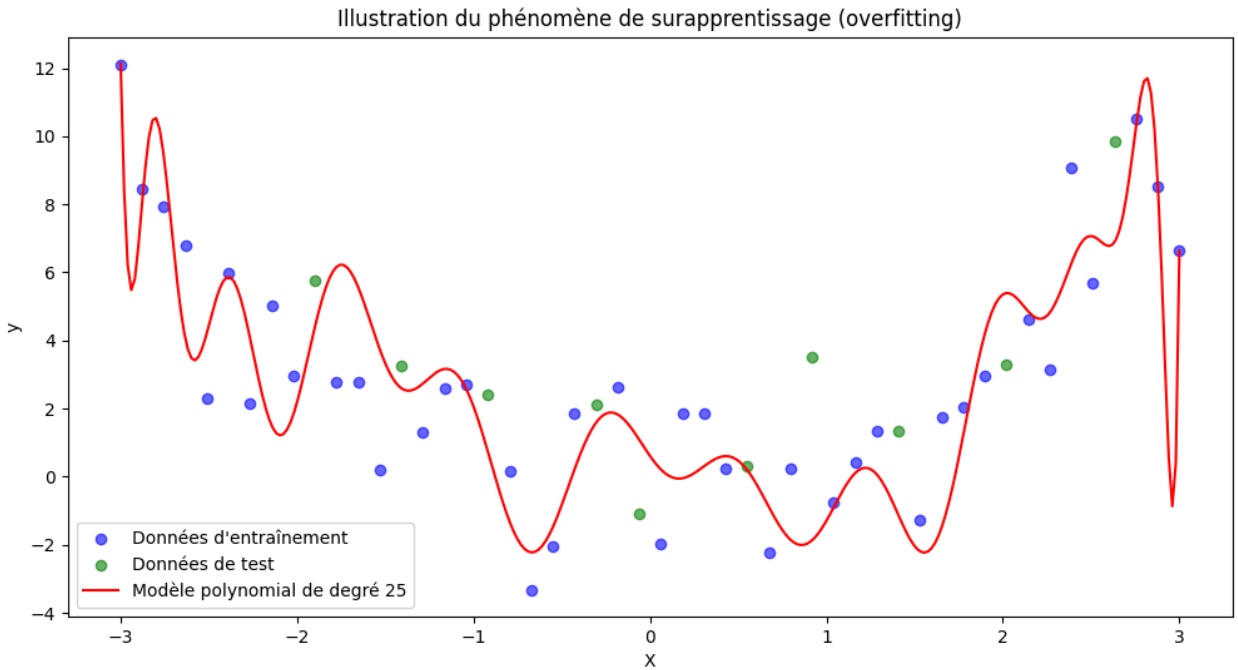
On dit qu'un modèle de machine learning est en régime de sous-apprentissage (underfitting) lorsqu'il n'arrive pas à capturer la complexité (l'information) présente dans le jeu de données d'entraînement.



Sur-apprentissage

Definition

On dit qu'un modèle de machine learning est en régime de sur-apprentissage (overfitting) lorsqu'il n'arrive pas à généraliser à des données non encore observées, i.e. lorsqu'il est trop adapté aux données d'entraînement.



Compromis biais-variance

Supposons $Y = f^\star(X) + \varepsilon$, avec $\mathbb{E}[\varepsilon \mid X] = 0$ et $\mathrm{Var}(\varepsilon \mid X) = \sigma^2$. Pour un point x fixé :

$$\mathbb{E}_{\mathcal{D},Y}\left[(\widehat{f}_{\mathcal{D}}(x) - Y)^2\right] = \underbrace{\left(\mathbb{E}_{\mathcal{D}}[\widehat{f}_{\mathcal{D}}(x)] - f^\star(x)\right)^2}_{\text{biais}^2} + \underbrace{\mathrm{Var}_{\mathcal{D}}(\widehat{f}_{\mathcal{D}}(x))}_{\text{variance}} + \underbrace{\sigma^2}_{\text{bruit irréductible}}.$$

La régularisation permet de réduire la variance en contraignant la complexité du modèle, au prix d'une augmentation possible du biais.

Métriques de performance : régression

On dispose d'un certain nombre de métriques pour évaluer les performances des modèles de machine learning. Celles-ci peuvent être divisées en deux catégories.

Régression

- L'erreur quadratique moyenne (MSE) : elle est définie comme la moyenne des carrés des écarts entre les prédictions et les valeurs observées.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2$$

- La racine carrée de l'erreur quadratique moyenne (RMSE) :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2}$$

Métriques de performance : classification 1/2

Accuracy : l'accuracy est la proportion de prédictions correctes. Elle est définie comme :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}}$$

Matrice de confusion : La matrice de confusion est une représentation permettant d'offrir plus de finesse par rapport à l'accuracy, notamment quand le jeu de données est déséquilibré (présence de classes majoritaires). Elle compare les prédictions du modèle avec les valeurs réelles et est structurée comme suit :

		Valeur Prédite	
		Positif	Négatif
Valeur Réelle	Positif	Vrai Positif (VP)	Faux Négatif (FN)
	Négatif	Faux Positif (FP)	Vrai Négatif (VN)

Métriques de performance : classification 2/2

À partir de la matrice de confusion, on peut dériver d'autres métriques :

- Précision : elle est définie comme la proportion des prédictions correctes parmi toutes les prédictions positives :

$$\text{Précision} = \frac{VP}{VP + FP}$$

- Rappel (recall) : il représente la proportion des vrais positifs correctement prédits par le modèle.

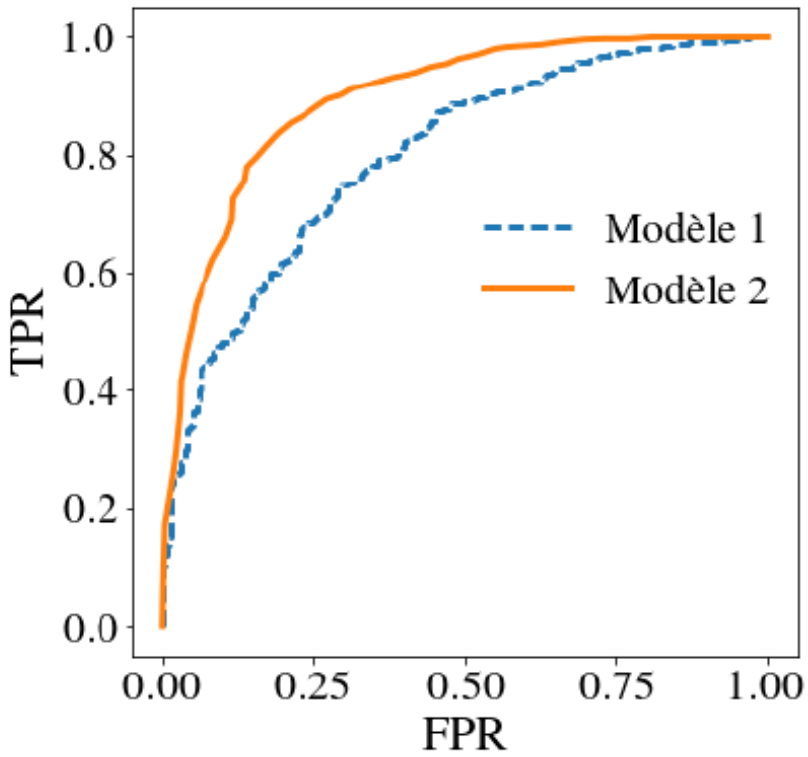
$$\text{Rappel} = \frac{VP}{VP + FN}$$

- Score F1 (F1-score) : Le score F1 est défini comme la moyenne harmonique de la précision et du rappel.

$$\text{Score F1} = 2\frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Métriques de performance : courbe ROC

- **Courbe ROC (Receiver Operating Characteristic)** : elle représente le taux de vrais positifs $TPR = VP/(VP + FN)$ en fonction du taux de faux positifs $FPR = FP/(FP + VN)$ lorsque le seuil varie.
- L'aire sous la courbe (ROC-AUC) mesure la capacité de classement du score, mais ne fixe pas le seuil de décision.



Classes déséquilibrées

L'accuracy et la ROC-AUC peuvent être insuffisantes lorsque la classe positive est rare.

- La courbe précision–rappel montre le compromis entre la précision et le rappel lorsque le seuil varie.
- La PR-AUC dépend de la prévalence de la classe positive ; elle doit être comparée à cette prévalence.
- En multiclasse, il faut préciser l'agrégation : micro, macro ou pondérée.
- Le seuil final doit être choisi à partir du coût relatif des faux positifs et des faux négatifs.

Calibration des probabilités

Un score bien classé n'est pas nécessairement une probabilité bien calibrée.

- Une prédiction de $0{,}8$ est calibrée si l'événement survient environ $80\,\%$ du temps parmi les cas comparables.
- La courbe de calibration compare fréquences observées et probabilités annoncées.
- Le score de Brier et la log-loss évaluent conjointement exactitude et confiance.
- Une calibration doit être ajustée sur des données distinctes de celles utilisées pour entraîner le modèle.

Choix du seuil de décision

Pour un score probabiliste $\widehat{p}(x)$, la règle de décision s'écrit

$$\widehat{y}_t(x) = \mathbf{1}\{\widehat{p}(x) \geq t\}.$$

Le seuil $t = 0{,}5$ n'est pas une règle universelle.

- Diminuer t augmente généralement le rappel, mais aussi le nombre de faux positifs.
- Le seuil peut être choisi sur le jeu de validation en minimisant

  $$C_{FP}\, FP(t) + C_{FN}\, FN(t),$$

  où C_{FP} et C_{FN} représentent les coûts des deux types d'erreurs.
- Si $\widehat{p}(x)$ est calibrée et si les bonnes décisions ont un coût nul, le seuil théorique est

  $$t^* = \frac{C_{FP}}{C_{FP} + C_{FN}}.$$

- Le seuil retenu doit être réévalué si les coûts ou la prévalence changent.

Sélection de modèle

Pour sélectionner le modèle le plus pertinent par rapport à une métrique donnée, on applique la méthodologie suivante :

- On partitionne le jeu de données disponible en trois parties : un jeu d'entraînement, un jeu de validation et un jeu de test.
- On entraîne M modèles sur le jeu d'entraînement.
- On évalue les performances respectives des M modèles sur le jeu de validation et on sélectionne le meilleur.
- Le modèle sélectionné est ensuite évalué sur le jeu de test. Idéalement, le jeu de test est ainsi utilisé une seule fois.

Réglage des hyperparamètres

Les hyperparamètres sont choisis à partir des performances de validation, jamais à partir du jeu de test.

- **Recherche sur grille** : évalue toutes les combinaisons d'une grille finie. Elle est simple, mais son coût croît rapidement avec le nombre de paramètres.
- **Recherche aléatoire** : tire des configurations dans des distributions fixées. À budget égal, elle explore souvent davantage de valeurs utiles lorsque seuls quelques paramètres sont déterminants.
- **Optimisation bayésienne** : construit un modèle probabiliste des performances observées, puis choisit les essais suivants à l'aide d'une fonction d'acquisition.

Le budget de calcul, les domaines de recherche et la métrique doivent être fixés avant de consulter le jeu de test.

Validation croisée imbriquée

Une validation croisée simple devient optimiste si elle sert à la fois au réglage et à l'évaluation du modèle.

1. La boucle **externe** réserve successivement un pli pour l'évaluation.
2. Dans chaque ensemble d'entraînement externe, la boucle **interne** sélectionne les hyperparamètres.
3. Le modèle réglé est réentraîné sur l'ensemble d'entraînement externe, puis évalué sur le pli externe resté intact.
4. Les scores externes sont agrégés pour estimer la performance de toute la procédure de sélection.

Après cette estimation, une configuration finale peut être choisie sur les données de développement et évaluée une seule fois sur un éventuel jeu de test indépendant.

Éviter les fuites de données

Le jeu de test doit rester indépendant de toutes les décisions prises pendant le développement.

- Ajuster la standardisation, l'imputation, la sélection de variables et la réduction de dimension uniquement sur les données d'entraînement.
- Regrouper ces opérations et le modèle dans une même chaîne de traitement.
- Utiliser une validation imbriquée lorsque la validation croisée sert à la fois au réglage des hyperparamètres et à l'estimation de la performance.
- Adapter le découpage aux données : stratification, groupes indépendants ou ordre temporel.

Validation croisée k-fold

La validation croisée permet d'estimer la performance d'une procédure d'apprentissage sur plusieurs découpages des données. Deux variantes classiques sont la validation k-fold et le Leave-One-Out (LOO).

La validation croisée k-fold s'effectue selon la méthodologie suivante :

- On partitionne le jeu de données $\mathcal{D}$ en K parties ayant approximativement la même taille.
- Pour chaque partie $\mathcal{D}_k$, on entraîne le modèle sur l'ensemble des données restantes $\bigcup_{i \neq k} \mathcal{D}_i$. Ce modèle est ensuite évalué sur $\mathcal{D}_k$.
- On considère la moyenne des performances du modèle sur les différents $\mathcal{D}_k$.

La dispersion des scores entre plis renseigne sur la sensibilité au découpage, mais elle ne constitue pas directement un intervalle de confiance indépendant.

Validation croisée Leave-One-Out

La validation croisée Leave-One-Out est un cas particulier de la validation croisée k-fold où le nombre de plis est égal au nombre d'observations ($K = n$).

La validation croisée LOO s'effectue selon la méthodologie suivante :

- Chaque pli de validation contient une observation.
- Pour chaque observation, on entraîne le modèle sur les $n - 1$ autres observations et on l'évalue sur l'observation laissée de côté.
- On agrège les n erreurs obtenues.

LOO utilise presque toutes les données pour chaque apprentissage et présente généralement un faible biais. En revanche, son coût est élevé et son estimation peut avoir une variance importante, car les ensembles d'entraînement sont très corrélés.

Bootstrap

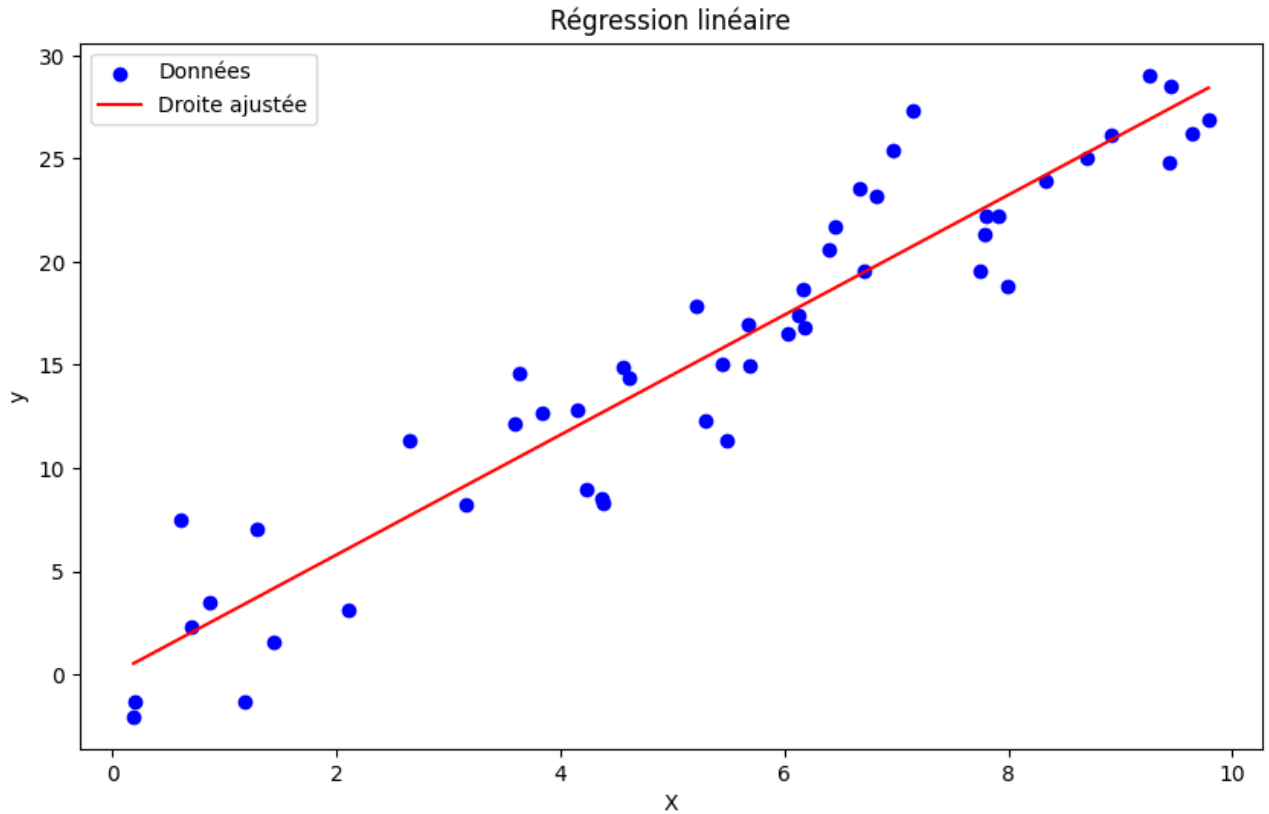
Le bootstrap construit B échantillons de taille n en tirant avec remise dans le jeu de données $\mathcal{D}$. Ces échantillons se chevauchent : il ne s'agit pas d'une partition.

Pour évaluer un modèle, on peut utiliser les observations absentes de chaque échantillon bootstrap, dites *out-of-bag*. Un échantillon laisse en moyenne environ $36{,}8\,\%$ des observations hors échantillon. Les variantes .632 corrigent en partie le biais de cette estimation.

Régression linéaire simple

Soit un ensemble de n observations $x_1, \ldots, x_n$ avec les réponses correspondantes $y_1, \ldots, y_n$. On cherche le modèle linéaire qui ajuste le mieux ces données.

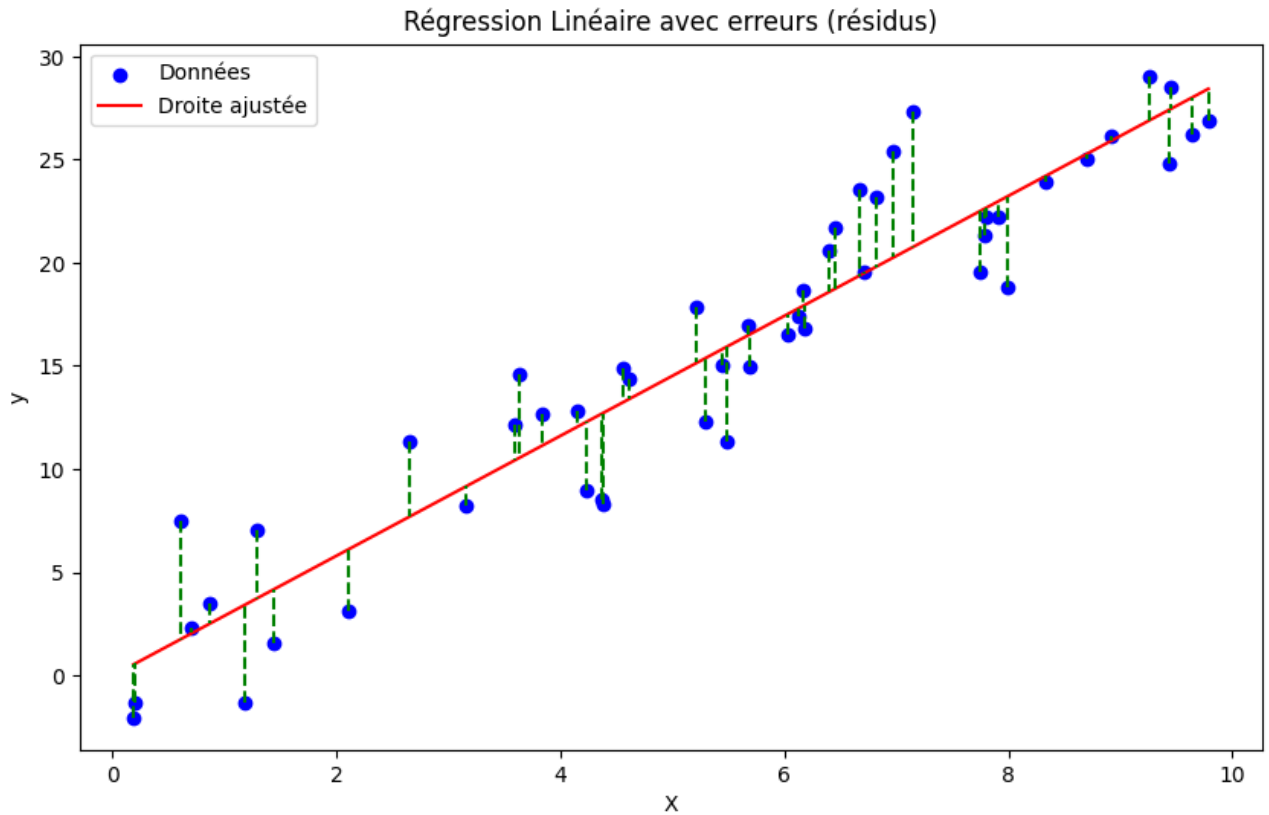
$$\hat{y} = \beta_0 + \beta_1 x$$



Résidus

Definition

Soit $\hat{y}_i = \beta_0 + \beta_1 x_i$ la i-ième prédiction du modèle. Le i-ième résidu est $e_i = y_i - \hat{y}_i$.



Méthode des moindres carrés

On considère la somme des carrés des résidus, notée RSS :

$$RSS = e_1^2 + e_2^2 + ... + e_n^2$$

La méthode des moindres carrés estime les coefficients β_0 et β_1 en minimisant la RSS :

$$\arg\min_{\beta_0,\beta_1} \left(\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 \right)$$

Le problème peut être résolu d'une manière analytique. On obtient :

$$\begin{aligned} \beta_1 &= \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \\ \beta_0 &= \bar{y} - \beta_1 \bar{x} \end{aligned} \tag{1}$$

Régression linéaire multiple

On considère n observations $X^1, X^2, ..., X^n$ où chaque observation X^i est désormais un vecteur ayant p composantes (p variables explicatives).

$$X^i = \begin{pmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_p^i \end{pmatrix}$$

La régression linéaire s'écrit alors :

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

Les coefficients $\beta_0, \beta_1, \ldots, \beta_p$ sont déterminés par la méthode des moindres carrés :

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\arg\min} \sum_{i=1}^{n} \left(y^i - (\beta_0 + \sum_{j=1}^{p} \beta_j x_j^i) \right)^2$$

L'équation normale

La régression linéaire peut être écrite sous une forme vectorielle :

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \qquad \widehat{\mathbf{y}} = \mathbf{X}\widehat{\boldsymbol{\beta}}.$$

où $\mathbf{X}$ est la matrice de conception.

$$\mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \dots & x_p^1 \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_1^n & \dots & x_p^n \end{pmatrix}$$

La somme des carrés des résidus est donnée par :

$$RSS = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

On peut montrer que si la matrice de conception $\mathbf{X}$ est de rang colonne plein, alors :

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.$$

Régression polynomiale

La régression polynomiale reste linéaire par rapport aux coefficients. Pour une variable x et un degré d :

$$\widehat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d.$$

Avec plusieurs variables, on peut ajouter des puissances et des interactions, puis ajuster les coefficients par moindres carrés.

Remarque : On peut également utiliser d'autres fonctions de base, par exemple $\cos(x)$ ou $\log(x)$. Le modèle reste linéaire par rapport aux coefficients.

Choix de modèle

Le coefficient de détermination R^2 mesure l'ajustement sur l'échantillon :

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

Le coefficient ajusté pénalise l'ajout de variables :

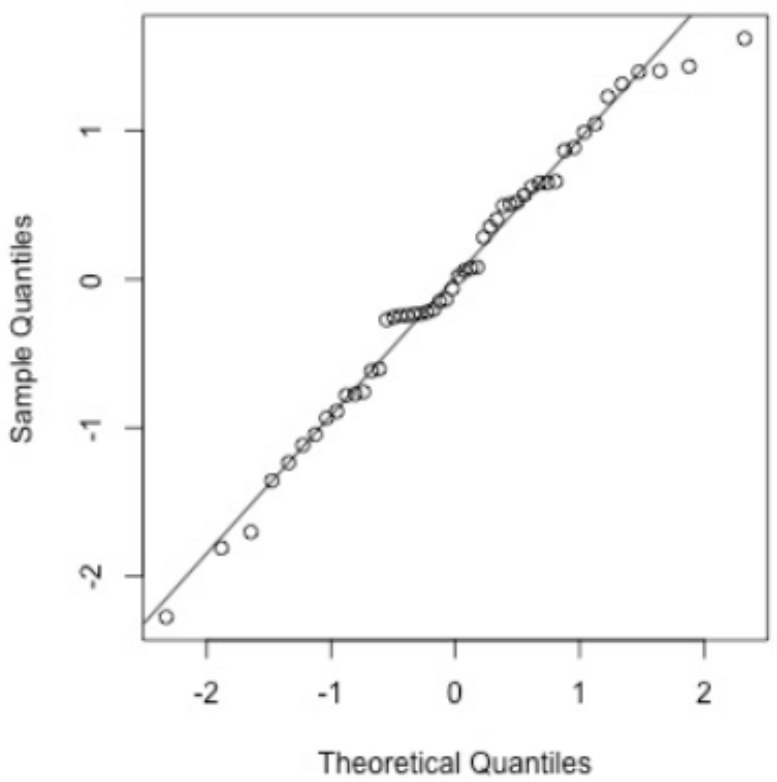
$$R^2_{\text{aj}} = 1 - (1 - R^2)\frac{n-1}{n-p-1},$$

où p est le nombre de variables explicatives.

La performance prédictive doit être évaluée sur validation ou par validation croisée. AIC et BIC comparent des modèles probabilistes ajustés sur les mêmes données, sous leurs hypothèses de vraisemblance.

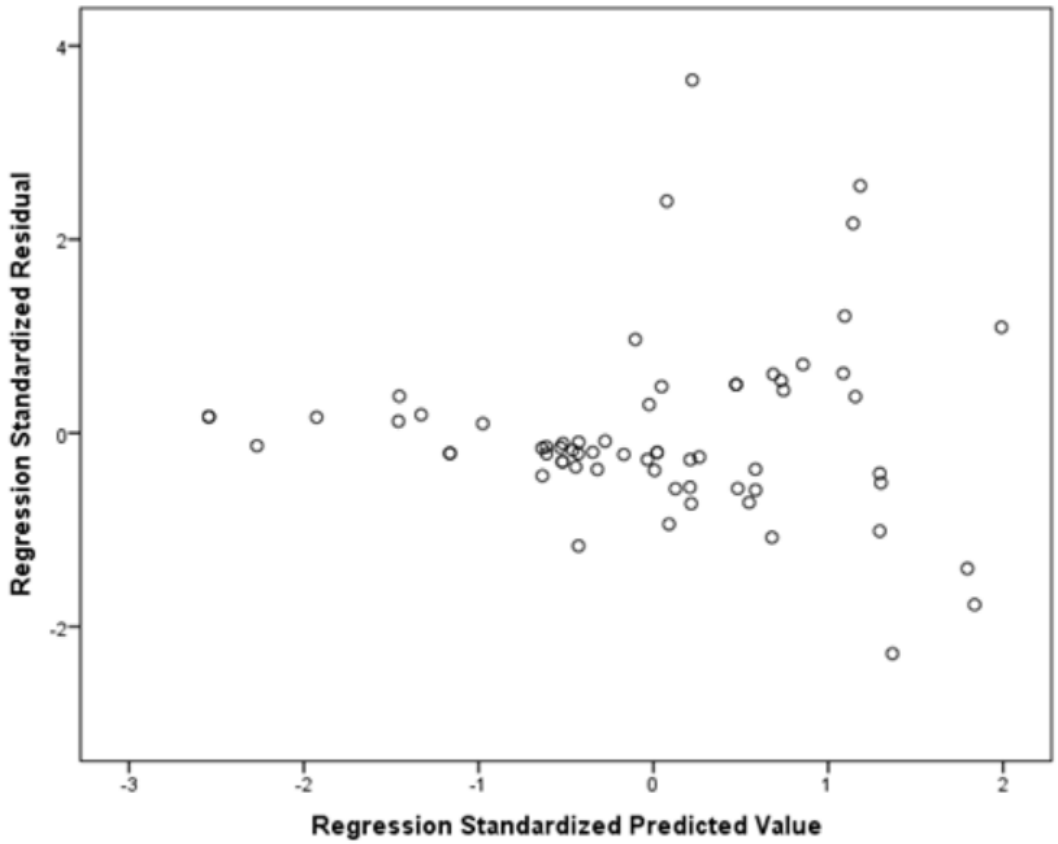
Diagnostic de la régression linéaire : normalité des résidus

Le diagramme quantile–quantile évalue l'hypothèse de normalité des résidus. Cette hypothèse intervient surtout dans les tests et intervalles de confiance ; elle n'est pas nécessaire à l'absence de biais des moindres carrés.



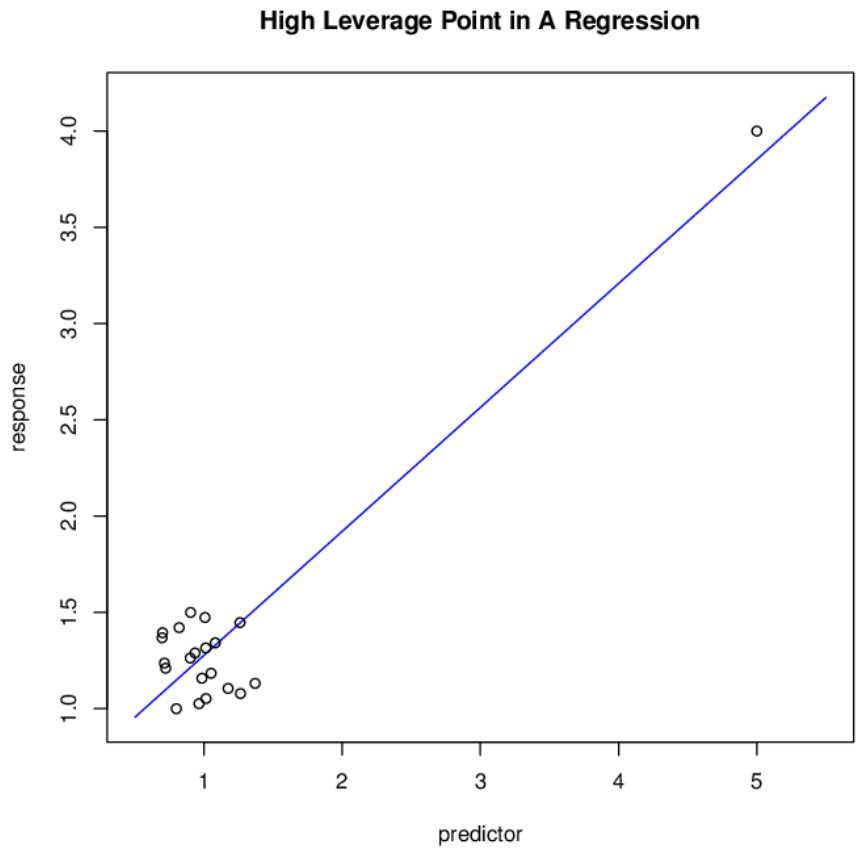
Diagnostic de la régression linéaire : homoscédasticité

Le nuage des résidus en fonction des valeurs ajustées doit rester sans structure et présenter une dispersion approximativement constante. Une forme en éventail indique une hétéroscédasticité.



Diagnostic de la régression linéaire : effet levier

Un point à fort levier possède des valeurs explicatives éloignées du centre des données. Son influence dépend aussi de son résidu ; la distance de Cook combine ces deux effets.



Régression Ridge

La régularisation Ridge, dite également régularisation L_2, limite l'amplitude des coefficients lors de la minimisation du risque empirique.

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\arg\min} \left(\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right)$$

sous la contrainte :

$$\sum_{j=1}^{p} \beta_j^2 \leq C$$

Le problème peut être écrit d'une manière plus compacte :

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\arg\min} \left(\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right)$$

L'intercept β_0 n'est pas pénalisé. Le paramètre λ contrôle la force de la régularisation ; il est choisi sur les données de validation.

Formulation vectorielle de la régression Ridge 1/2

Après centrage de $\mathbf{X}$ et de $\mathbf{y}$, l'intercept est traité séparément et n'est pas pénalisé. Le critère Ridge s'écrit :

$$RSS = (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

Après centrage des variables, les coefficients peuvent être calculés en minimisant ce critère.

$$\arg\min_{\boldsymbol{\beta}}(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

On peut alors montrer que :

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

où $\mathbf{I}$ est la matrice identité de dimension $p \times p$.

Formulation vectorielle de la régression Ridge 2/2

La matrice de conception $\mathbf{X} \in \mathbb{R}^{n \times p}$ admet la SVD réduite :

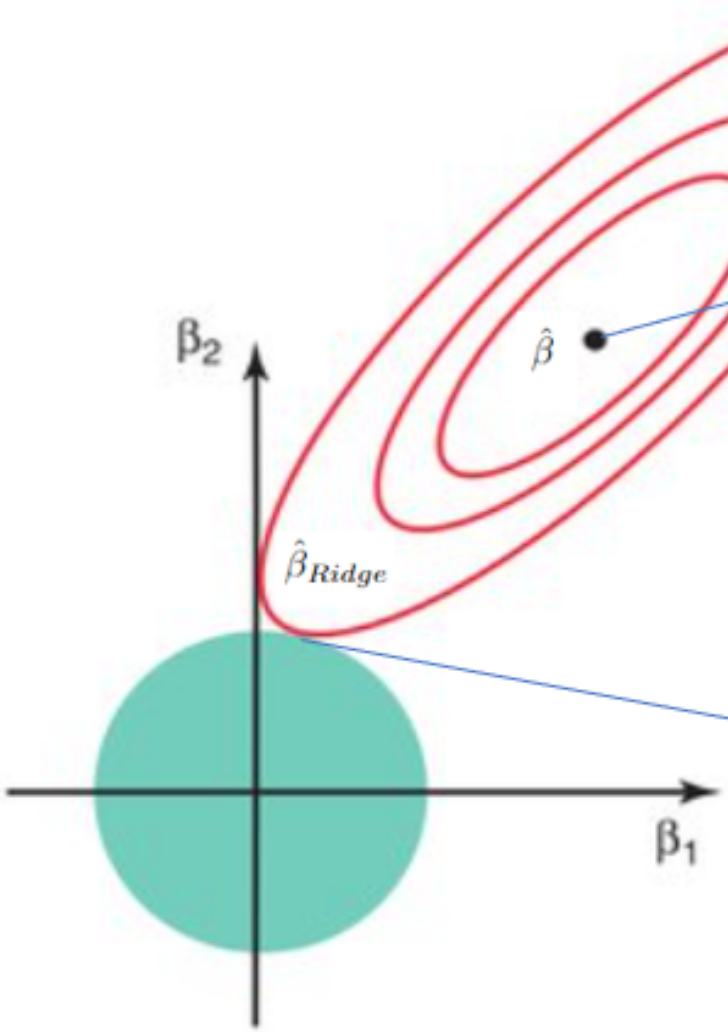
$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

où $\mathbf{U} \in \mathbb{R}^{n \times r}$ et $\mathbf{V} \in \mathbb{R}^{p \times r}$ ont des colonnes orthonormées, et $r = \mathrm{rang}(\mathbf{X})$.
On peut montrer que :

$$\begin{aligned} \mathbf{X}\widehat{\boldsymbol{\beta}} &= \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\boldsymbol{y} \\ &= \sum_{j=1}^{r} \frac{d_j^2}{d_j^2 + \lambda} u_j u_j^T \mathbf{y} \end{aligned} \tag{2}$$

Ridge atténue principalement les directions associées aux petites valeurs singulières.

Interprétation géométrique de régression Ridge



$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\arg\min} \left(\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{i=1}^{p} x_{ij}\beta_j)^2 \right)$$

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\arg\min} \left(\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{i=1}^{p} x_{ij}\beta_j)^2 \right)$$

$$\sum_{j=1}^{p} \beta_j^2 \leq Cte$$

Régression Lasso

La régularisation Lasso, dite également régularisation L_1, favorise des coefficients exactement nuls et produit ainsi des modèles parcimonieux. En présence de variables fortement corrélées, la sélection peut cependant être instable.

La régression Lasso est formalisée de la manière suivante :

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\arg\min} \left(\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \right)$$

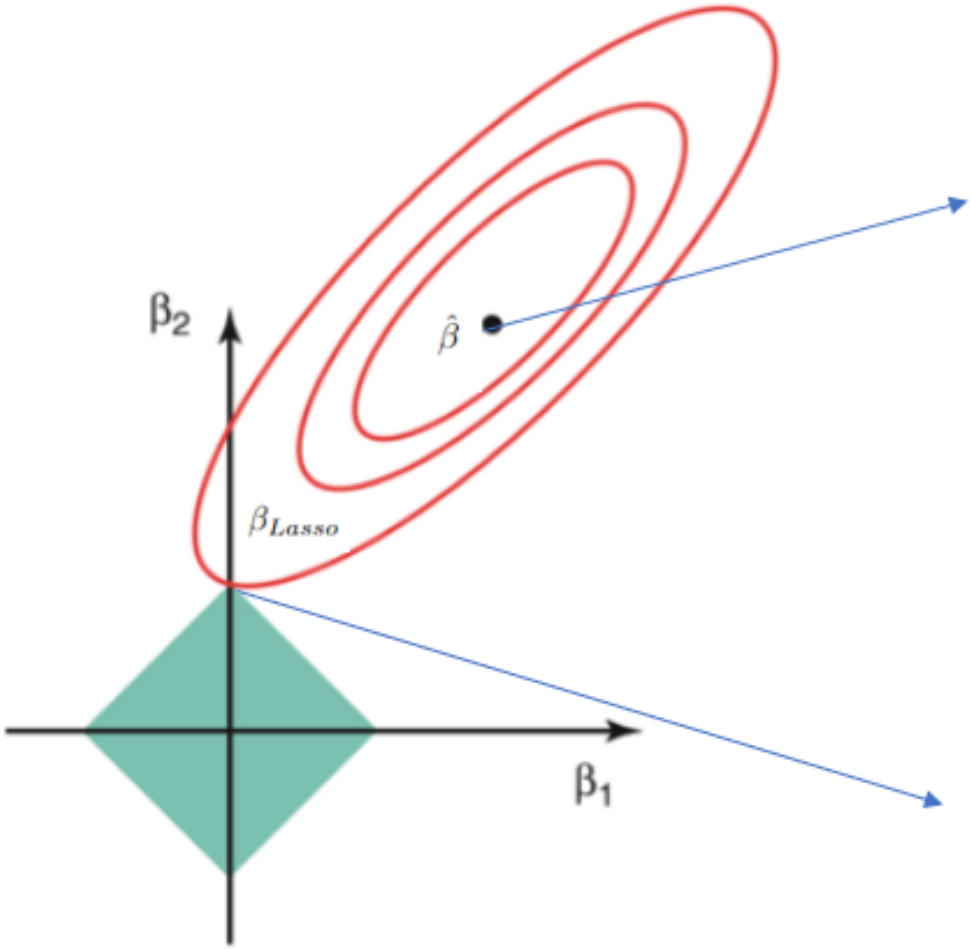
sous la contrainte :

$$\sum_{j=1}^{p} |\beta_j| \leq C$$

ou, sous forme pénalisée :

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\arg\min} \left(\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right)$$

Interprétation géométrique de régression Lasso



$$\arg\min_{\beta_0,\beta_1,\ldots,\beta_p} \left(\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{i=1}^{p} x_{ij}\beta_j)^2 \right)$$

$$\arg\min_{\beta_0,\beta_1,\ldots,\beta_p} \left(\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{i=1}^{p} x_{ij}\beta_j)^2 \right)$$

$$\sum_{j=1}^{p} |\beta_j| \leq Cte$$

Elastic net

Elastic Net combine les pénalités Ridge et Lasso avec un paramètre $\alpha \in [0, 1]$.

Le problème s'écrit ainsi :

$$\underset{\beta_0, \boldsymbol{\beta}}{\arg\min} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2$$

sous la contrainte :

$$\alpha \sum_{j=1}^{p} |\beta_j| + (1 - \alpha) \sum_{j=1}^{p} \beta_j^2 \leq C$$

Ou

$$\underset{\beta_0, \boldsymbol{\beta}}{\arg\min} \left[\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + (1 - \alpha) \sum_{j=1}^{p} \beta_j^2 \right) \right]$$

Régression logistique : introduction

La régression logistique est une technique d'analyse statistique utilisée pour modéliser la probabilité d'une variable dépendante binaire. C'est un cas particulier de modèle linéaire généralisé qui est utilisé pour des problèmes de classification.

Principes de la régression logistique :

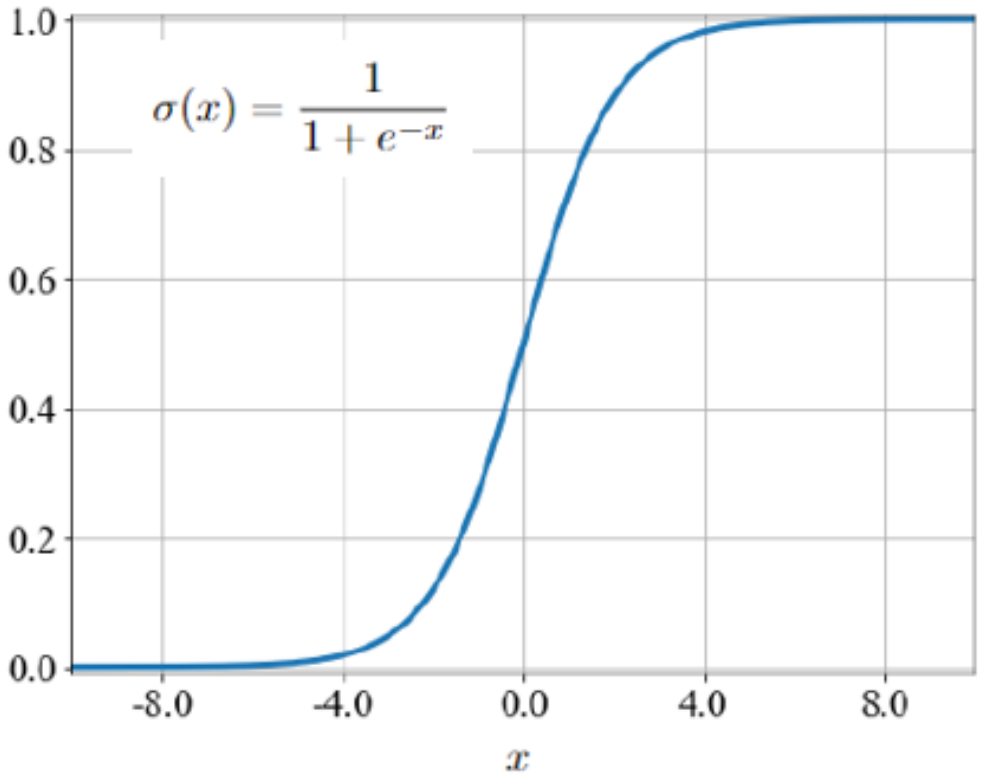
- **Variable réponse** : on modélise $P(Y = 1 \mid X = x)$ en fonction des variables explicatives x.
- **Cote** : le rapport $p(x)/(1 - p(x))$ est la cote de l'événement. Un *odds ratio* compare deux cotes.

$$\ln \frac{p(\mathbf{x})}{1 - p(\mathbf{x})} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$$

Régression logistique : fonction sigmoïde

Après quelques simplifications, on peut écrire la probabilité $p(x)$ (la probabilité pour que y soit un succès par exemple) :

$$p(\mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \boldsymbol{\beta}^T \mathbf{x})}}$$



Calcul des coefficients de la régression logistique

Les coefficients de la régression logistique peuvent être calculés en minimisant le risque empirique par rapport à une fonction de coût sous forme d'entropie croisée :

$$\arg\min_{\beta_0,\boldsymbol{\beta}} -\sum_{i=1}^{n} [y_i \log p_i + (1-y_i)\log(1-p_i)]$$

k plus proches voisins (k-NN)

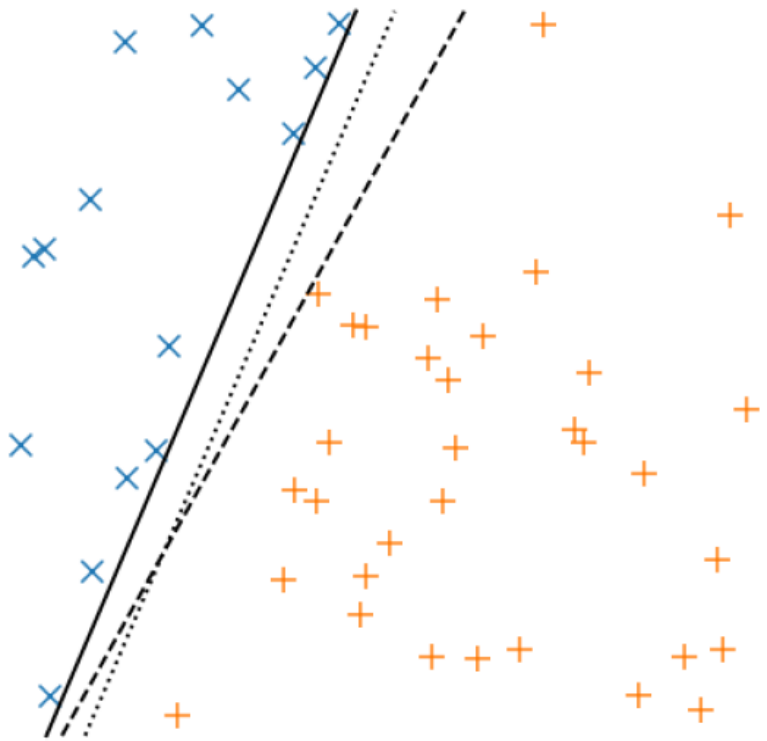
Pour prédire en un point x, la méthode des K plus proches voisins recherche les K observations d'entraînement les plus proches.

- En classification, la classe est déterminée par vote, éventuellement pondéré par la distance.
- En régression, la prédiction est une moyenne locale.
- Un petit K produit un modèle flexible et sensible au bruit ; un grand K augmente le biais.
- La méthode dépend fortement de la métrique, de l'échelle des variables et de la dimension.

Le choix de K et de la métrique s'effectue sur les données de validation.

Machines à vecteurs de support (SVM)

Considérons un problème de classification binaire. Dans le cas séparable, un SVM recherche l'hyperplan qui maximise la distance minimale aux observations d'entraînement. Les observations qui atteignent cette distance sont les vecteurs supports.

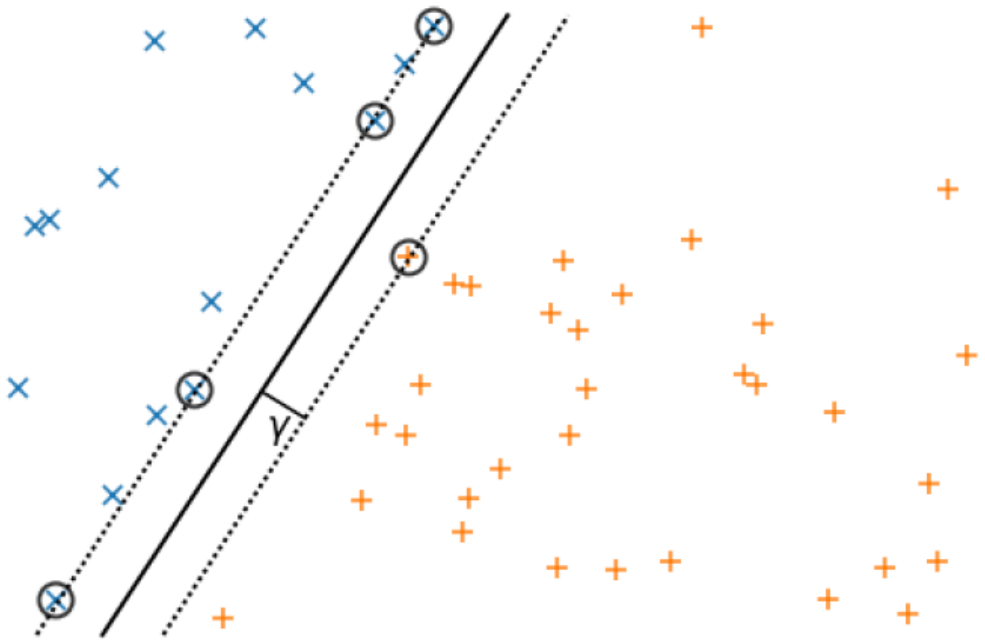


Séparation linéaire et marge

Soient n observations (x_i, y_i), avec $x_i \in \mathbb{R}^p$ et $y_i \in \{-1, +1\}$. Le vecteur $w \in \mathbb{R}^p$ contient un coefficient par variable explicative.

On cherche un score $h(x) = w^T x + b$ tel que $y_i h(x_i) \geq 1$ pour toutes les observations dans le cas séparable.

L'hyperplan d'équation $h(x) = 0$ est ainsi le plan séparateur pour les deux classes.



Formulation primale

La marge géométrique signée de l'observation (x_k, y_k) est donnée par :

$$\gamma_k = y_k \frac{w^T x_k + b}{\|w\|}$$

Les poids w et le biais b peuvent être calculés en résolvant le problème d'optimisation suivant :

$$\arg\max_{w,b} \left(\frac{1}{\|w\|} \min_k y_k (w^T x_k + b) \right)$$

En pratique, il est plus simple de résoudre le problème équivalent :

$$\arg\min_{w,b} \frac{1}{2} \|w\|^2$$

sous les contraintes :

$$y_k (w^T x_k + b) \geq 1, \qquad k = 1, \ldots, n.$$

Formulation duale 1/2

Le problème d'optimisation peut être résolu à l'aide des multiplicateurs de Lagrange α. Le lagrangien s'écrit :

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 + \sum_{k=1}^{n} \alpha_k \left[1 - y_k(w^T x_k + b)\right].$$

En minimisant le lagrangien par rapport à w et b

$$\begin{aligned} \nabla_w L(w, b, \alpha) &= 0 \\ \frac{\partial L(w, b, \alpha)}{\partial b} &= 0 \end{aligned} \tag{3}$$

on obtient :

$$\begin{cases} \sum_{k=1}^{n} \alpha_k y_k x_k = w \\ \sum_{k=1}^{n} \alpha_k y_k = 0 \end{cases}$$

Formulation duale 2/2

En injectant les équations précédentes dans la formule du lagrangien, on obtient :

$$g(\alpha) = \sum_{k=1}^{n} \alpha_k - \frac{1}{2} \sum_{k=1}^{n} \sum_{j=1}^{n} \alpha_k \alpha_j y_k y_j x_k^T x_j.$$

Le problème dual s'écrit ainsi :

$$\arg\max_{\alpha} \left(\sum_{k=1}^{n} \alpha_k - \frac{1}{2} \sum_{k=1}^{n} \sum_{j=1}^{n} y_k y_j \alpha_k \alpha_j x_k^T x_j \right)$$

sous les contraintes

$$\begin{cases} \alpha_k \geq 0, \ k = 1, 2, \ldots, n \\ \sum_{k=1}^{n} \alpha_k y_k = 0 \end{cases}$$

Marge souple

Lorsque les classes ne sont pas parfaitement séparables, on introduit des variables d'écart $\xi_i \geq 0$:

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

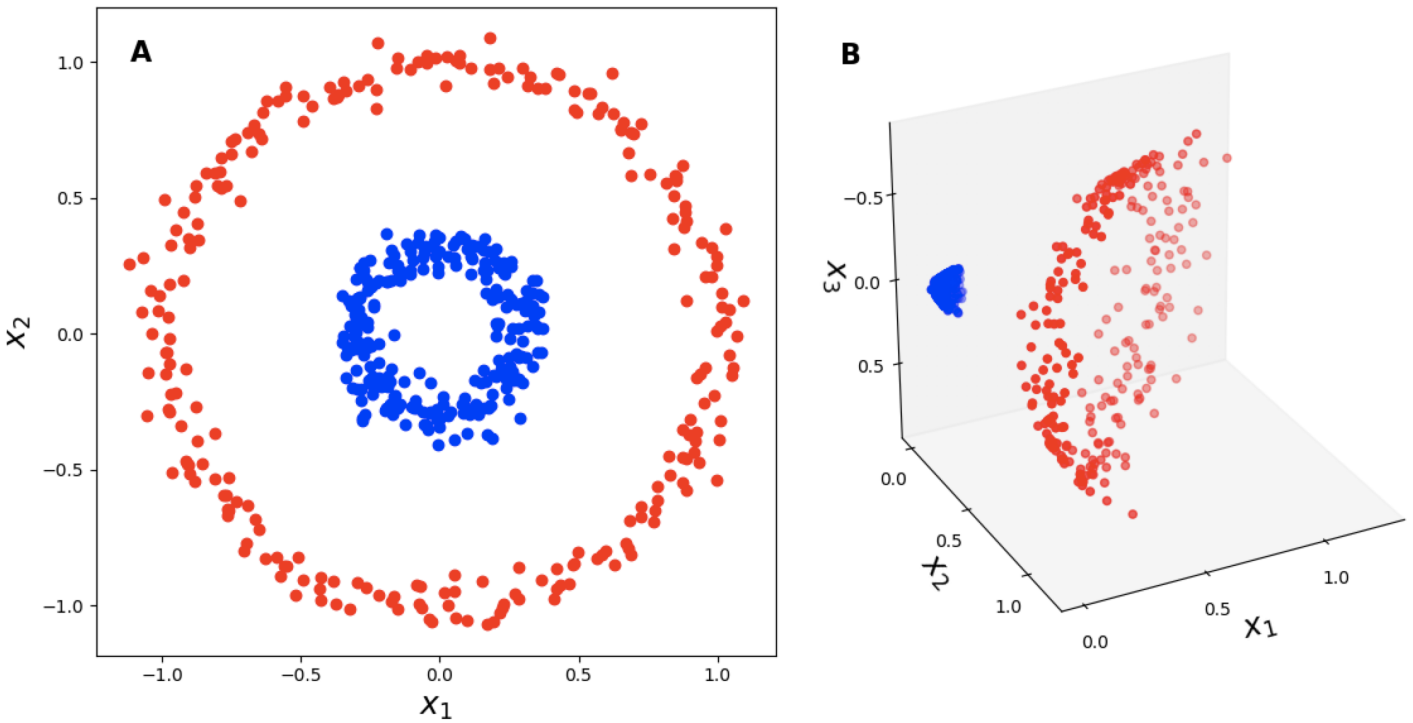
sous les contraintes

$$y_i(w^T x_i + b) \geq 1 - \xi_i.$$

Le paramètre C arbitre entre une marge large et les erreurs d'entraînement. Il doit être choisi par validation.

SVM à noyau

Pour traiter une séparation non linéaire, on remplace le produit scalaire $x_i^T x_j$ par un noyau $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$. La transformation Φ n'a pas besoin d'être calculée explicitement.



Noyaux positifs

Definition

Un noyau positif est une fonction K qui peut s'écrire comme un produit scalaire dans un espace de caractéristiques :

$$K(x, z) = \Phi(x) \cdot \Phi(z)$$

où Φ est généralement une fonction non linéaire.

Remarque : il n'est pas nécessaire de connaître explicitement la fonction Φ.

Théorème

Une fonction symétrique $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est un noyau positif si, pour tout n, tous points $x_1, \ldots, x_n$ et tous réels $c_1, \ldots, c_n$,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0.$$

La matrice de Gram $[K(x_i, x_j)]_{ij}$ est donc positive semi-définie.

Formulation duale des SVM à noyau

Le problème dual pour les SVM à noyau s'écrit alors :

$$\arg\max_{\alpha} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \right]$$

sous les contraintes

$$\begin{cases} 0 \leq \alpha_k \leq C, \quad k = 1, \ldots, n \\ \sum_{k=1}^{n} \alpha_k y_k = 0 \end{cases}$$

où $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$. Le paramètre C contrôle la pénalisation des violations de marge.

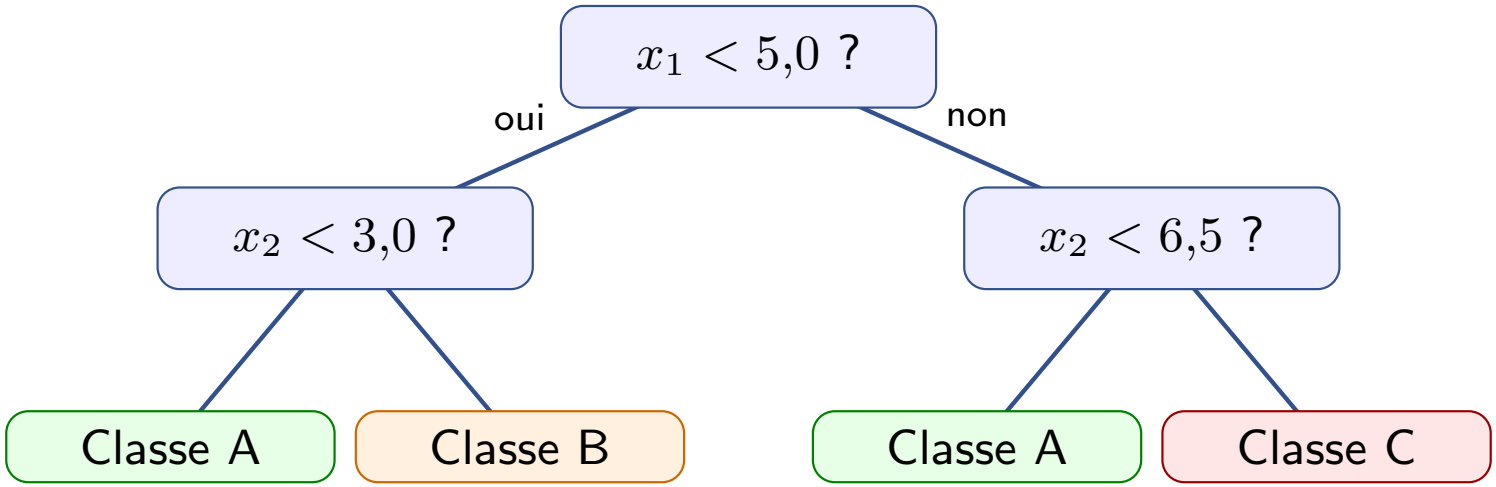
Exemples de noyaux

Les noyaux gaussien et polynomial sont deux choix classiques pour les SVM.

- Noyau gaussien : $K(x,z) = \exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$
- Noyau polynomial : $K(x,z) = (x^T z + 1)^d$, avec $d \in \mathbb{N}^*$.

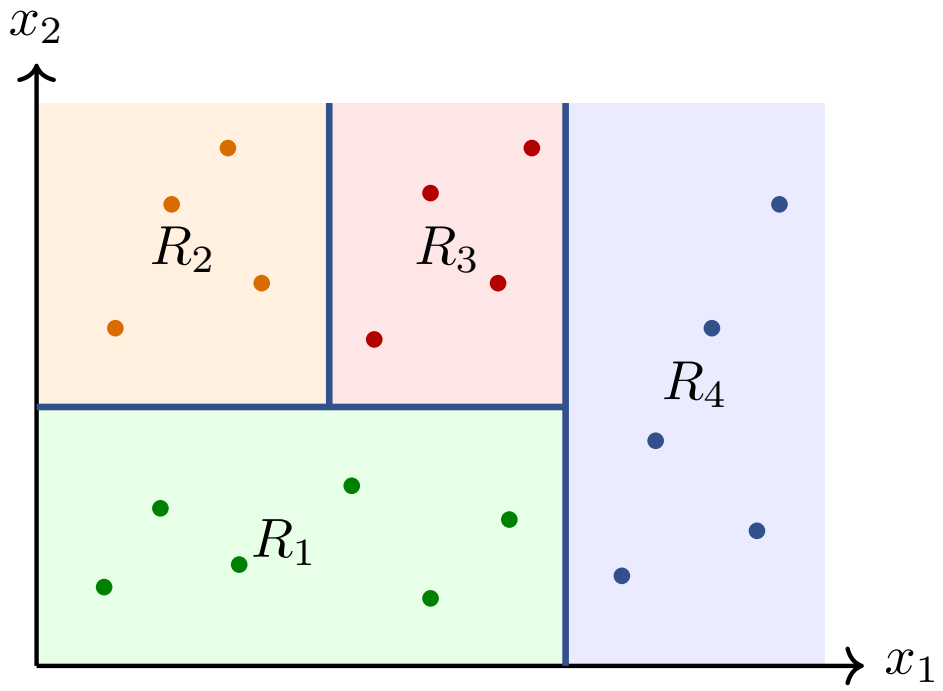
Arbre de décision

Un arbre de décision partitionne récursivement l'espace des variables à l'aide de règles simples. Chaque feuille porte une prédiction.



Entraînement des arbres de décision

Les arbres sont généralement entraînés avec CART (Classification and Regression Trees). Étant donné un ensemble d'observations (x_i, y_i), CART recherche à chaque nœud une variable et un seuil qui réduisent le plus l'impureté.



Coupures des variables continues

Pour une variable continue x_j, CART n'examine pas tous les seuils réels.

1. Trier les valeurs distinctes observées : $x_{(1)j} < \cdots < x_{(m)j}$.
2. Considérer les seuils situés entre deux valeurs consécutives, par exemple

$$s_k = \frac{x_{(k)j} + x_{(k+1)j}}{2}.$$

3. Pour chaque seuil, former $R_l = \{x : x_j \leq s_k\}$ et $R_r = \{x : x_j > s_k\}$.
4. Retenir la coupure qui produit la plus forte diminution du critère d'impureté.

Les variables continues peuvent ainsi être réutilisées à plusieurs niveaux de l'arbre avec des seuils différents.

Contrôle de la complexité des arbres

Un arbre développé jusqu'à obtenir des feuilles presque pures présente souvent une variance élevée.

- **Profondeur maximale** : limite le nombre de décisions successives.
- **Taille minimale d'une feuille** : évite les prédictions fondées sur trop peu d'observations.
- **Gain minimal** : refuse une coupure dont la réduction d'impureté est insuffisante.
- **Élagage** : construit un arbre large puis choisit un sous-arbre en minimisant

$$R_\alpha(T) = R(T) + \alpha\,|\mathcal{L}(T)|,$$

où $\mathcal{L}(T)$ est l'ensemble des feuilles.

Ces paramètres sont réglés sur les données de validation ou par validation croisée.

Arbres de régression

Pour une régression, on cherche la variable x_j et le seuil s minimisant la somme des carrés dans les deux régions :

$$\arg\min_{j,s} \Big(\sum_{x_i \in R_l(j,s)} (y_i - y_l(j,s))^2 + \sum_{x_i \in R_r(j,s)} (y_i - y_r(j,s))^2 \Big)$$

où $y_l(j,s)$ et $y_r(j,s)$ sont les moyennes des labels associées à chacune des deux régions formées par le partitionnement.

Il est important ici de noter que nous considérons comme critère d'optimisation la somme des erreurs quadratiques :

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

Arbres de classification

L'approche pour la classification est similaire à celle pour la régression ; la fonction de coût quadratique est remplacée cette fois par une fonction mesurant l'impureté au niveau des feuilles (le degré d'hétérogénéité des labels).
Pour un nœud R_m contenant N_m observations, on considère le problème suivant :

$$\arg\min_{j,s} \left[\frac{|R_l(j,s)|}{N_m} I(R_l(j,s)) + \frac{|R_r(j,s)|}{N_m} I(R_r(j,s)) \right]$$

On définit la proportion d'observations de la classe k dans un nœud R_m par :

$$p_{mk} = \frac{1}{|R_m|} \sum_{x_i \in R_m} \mathbf{1}\{y_i = k\}$$

Différentes fonctions mesurant l'impureté peuvent être considérées :

- Erreur de classification : $1 - \max_k p_{mk}$
- Indice de Gini : $\sum_{k=1}^{K} p_{mk}(1 - p_{mk})$
- Entropie : $-\sum_{k=1}^{K} p_{mk} \ln(p_{mk})$

Méthodes ensemblistes (bagging)

Le bagging vise principalement à réduire la variance en agrégeant des modèles ajustés sur des échantillons bootstrap.

- L'agrégation réduit la variance lorsque les erreurs des modèles ne sont pas parfaitement corrélées.
- Le bootstrap crée des jeux d'entraînement différents à partir du même échantillon initial.

Soit $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$. Le bagging suit les étapes suivantes :

- On construit B échantillons en bootstrap (tirage avec remise) à partir de $\mathcal{D}$.
- On entraîne un modèle sur chaque échantillon.
- On combine les prédictions issues des B modèles (vote majoritaire pour la classification et calcul de la moyenne pour la régression).

Forêts aléatoires (random forests)

La technique des forêts aléatoires (random forests) consiste à appliquer une approche de type bagging sur les arbres de décision.

L'algorithme random forests suit cette procédure :

- Tirer par bootstrap B échantillons de tailles n à partir de l'ensemble D.
- Pour chaque échantillon, développer un arbre jusqu'à atteindre un critère d'arrêt, par exemple une taille minimale $n_{\min}$ dans les nœuds.
  - Tirer d'une manière aléatoire m variables parmi les p variables.
  - Sélectionner la meilleure variable avec le meilleur point de partitionnement.
  - Partitionner le nœud en deux sous-branches.
- Agréger les arbres construits.

  Les prédictions sont agrégées selon qu'il s'agisse de régression ou de classification :

- Régression : moyenne $f^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$
- Classification : vote majoritaire.

Forêts aléatoires : propriétés

- Pour chaque observation, la prédiction OOB agrège uniquement les arbres dont l'échantillon bootstrap ne contenait pas cette observation.
- Le nombre m de variables candidates à chaque nœud contrôle la corrélation entre les arbres et doit être réglé.
- Si très peu de variables sont informatives parmi un grand nombre de variables, un m trop faible peut empêcher de les proposer aux nœuds.
- L'importance par diminution d'impureté est biaisée en faveur des variables comportant de nombreux seuils possibles. L'importance par permutation fournit souvent une lecture plus fiable.

Boosting

Le boosting construit un modèle additif en ajustant séquentiellement des apprenants faibles. Chaque nouvel apprenant dépend des erreurs ou des gradients produits par l'ensemble courant.

Principes de base :

- Construire séquentiellement des modèles faibles, généralement des arbres de décision.
- Chaque modèle suivant apporte une correction à l'ensemble déjà construit.
- Les contributions sont combinées par une somme pondérée ; la règle de pondération dépend de l'algorithme.

Paramètres importants :

- nombre et complexité des apprenants faibles ;
- taux d'apprentissage ;
- régularisation et sous-échantillonnage éventuel.

AdaBoost

AdaBoost construit un classificateur additif à partir de modèles faibles.

Algorithme :

1. Initialisez les poids des observations : $D_1(i) = \frac{1}{n}$.
2. Pour $t = 1, 2, ..., T$:
   - Entraînez un modèle faible $h_t(x)$.
   - Calculez l'erreur : $\epsilon_t = \sum_{i=1}^{n} D_t(i) \mathbf{1}[y_i \neq h_t(x_i)]$.
   - Calculez le poids du modèle : $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
   - Mettez à jour les poids : $D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$.
3. Modèle final : $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.

Remarques :

- Les formules supposent $y_i, h_t(x_i) \in \{-1, +1\}$.
- Dans le cas binaire standard, on requiert $\epsilon_t < 1/2$; sinon l'apprenant n'améliore pas le vote.
- AdaBoost ajuste les poids des observations pour se concentrer sur les erreurs difficiles, mais peut être sensible au bruit d'étiquetage.

Gradient boosting

Gradient boosting : Construction itérative d'un modèle additif en minimisant une fonction de perte.

Processus itératif :

1. Commencez avec un modèle initial : $f_0(x)$.
2. Pour $t = 1, 2, ..., T$:
   - Calculez les pseudo-résidus : $r_{ti} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}(x)}$.
   - Entraînez un modèle faible $h_t(x)$ sur r_{ti}.
   - Trouvez γ_t minimisant $\sum_i L(y_i, f_{t-1}(x_i) + \gamma h_t(x_i))$.
   - Mettez à jour : $f_t(x) = f_{t-1}(x) + \nu\gamma_t h_t(x)$, avec un taux $0 < \nu \leq 1$.
3. Modèle final : $f_T(x)$.

Remarques :

- Chaque modèle successif corrige les erreurs du modèle précédent.
- Les arbres de décision sont généralement utilisés comme modèles faibles.

Implémentations du gradient boosting

XGBoost, LightGBM et CatBoost reposent sur le même principe de modèle additif, mais diffèrent par leurs choix algorithmiques.

- **XGBoost** : régularisation explicite, approximation du second ordre et gestion efficace des valeurs manquantes.
- **LightGBM** : histogrammes et croissance feuille par feuille. La profondeur des feuilles doit être contrôlée.
- **CatBoost** : traitement natif des variables catégorielles par statistiques ordonnées et *ordered boosting*, afin de limiter les fuites liées à l'encodage.

Le choix dépend surtout de la taille des données, de la présence de variables catégorielles, des contraintes de calcul et du protocole de validation.

Apprentissage non supervisé

Apprentissage non supervisé

Dans l'apprentissage non supervisé, on cherche une structure, une représentation ou une distribution à partir d'observations sans variable cible.

On distingue notamment trois catégories :

- **Partitionnement** : regrouper les observations selon une notion explicite de similarité.
- **Réduction de dimension** : représenter les données dans un espace de dimension plus faible pour visualiser, compresser ou faciliter un apprentissage ultérieur.
- **Détection d'anomalies** : identifier des observations atypiques au regard d'une distribution ou d'un voisinage.

Méthode des k-moyennes (k-means)

Soit un ensemble de n observations $(x_1, x_2, \ldots, x_n)$. La méthode des k-means vise à former K groupes en minimisant l'inertie intra-groupes.

Plus formellement, le problème des k-means s'écrit :

$$\underset{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_K}{\arg\min} \sum_{k=1}^{K} \sum_{x \in \mathcal{C}_k} \|x - \mu_k\|^2$$

Ce problème est non convexe. L'algorithme de Lloyd fournit une solution locale qui dépend de l'initialisation.

Algorithme de Lloyd

Considérons n observations $x_1, x_2, \ldots, x_n$ et un nombre déterminé K de clusters, l'algorithme de Lloyd consiste à exécuter les étapes suivantes :

- On initialise K centroïdes, de préférence avec k-means++.
- On affecte chaque observation x_i au cluster dont le centroïde est plus proche.

$$k(x_i) = \underset{k=1,2,\ldots,K}{\arg\min} \|x_i - \mu_k\|$$

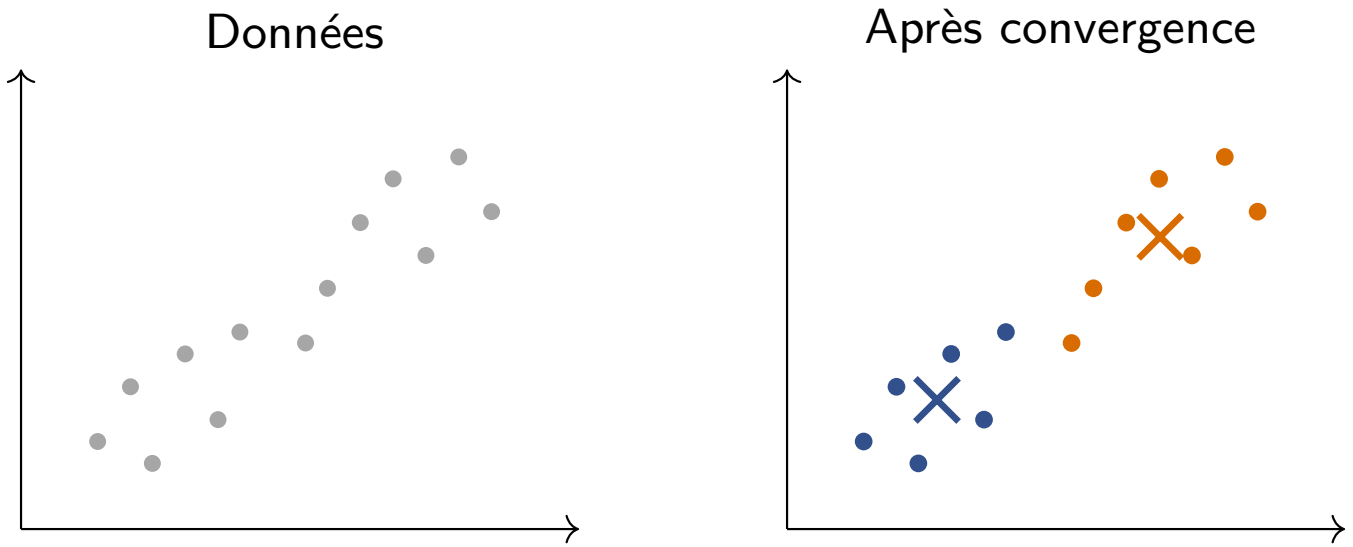
- On recalcule les centroïdes à partir de la nouvelle affectation.

$$\mu_k = \frac{1}{|\mathcal{C}_k|} \sum_{x_i \in \mathcal{C}_k} x_i$$

- On répète la procédure jusqu'à convergence (jusqu'à ce que les centroïdes soient stables).

Affectation aux groupes

L'algorithme de Lloyd alterne affectation et mise à jour des centroïdes. Il diminue l'inertie intra-classe à chaque itération et converge vers un minimum local.



K-means : limites

- Les variables doivent être placées sur des échelles comparables lorsque leurs unités ne traduisent pas directement leur importance.
- K-means est sensible aux valeurs aberrantes, qui peuvent déplacer fortement les centroïdes.
- Le résultat dépend de l'initialisation ; k-means++ et plusieurs redémarrages réduisent ce problème.
- K-means minimise des distances euclidiennes aux moyennes. K-medoids accepte des dissimilarités plus générales ; kernel k-means effectue le partitionnement dans un espace de caractéristiques.
- En grande dimension, les distances peuvent devenir peu discriminantes. Une réduction de dimension ou une autre notion de similarité peut être nécessaire.

Choix du nombre de groupes

Le nombre K ne doit pas être choisi à partir d'un seul graphique.

- La courbe d'inertie recherche un coude, parfois peu marqué.
- Le coefficient silhouette compare cohésion interne et séparation des groupes.
- La stabilité sous rééchantillonnage vérifie que les groupes ne dépendent pas excessivement des observations.
- Le choix final doit rester compatible avec l'usage et l'interprétation des groupes.

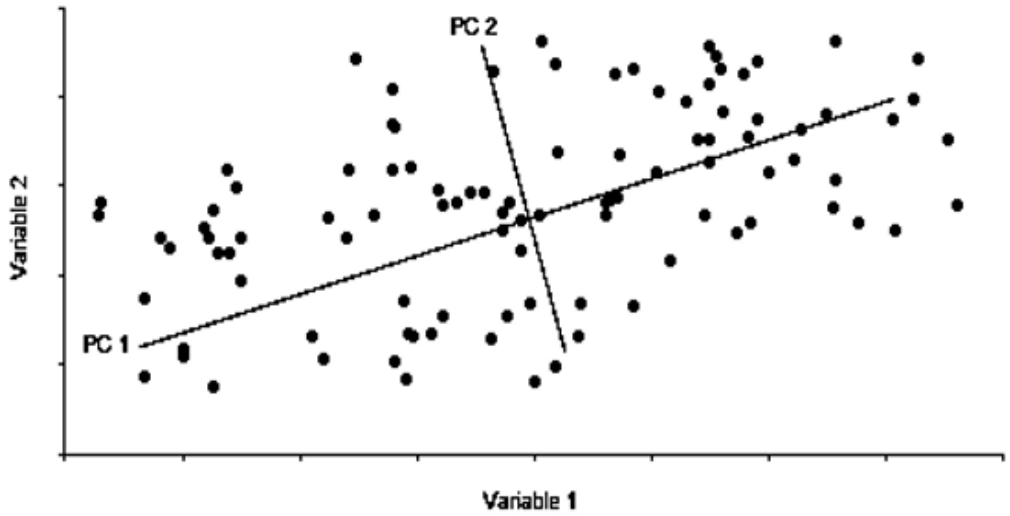
Autres méthodes de partitionnement

- **Classification hiérarchique** : construit une hiérarchie de groupes visualisée par un dendrogramme.
- **DBSCAN** : identifie des groupes de forme non convexe et marque les points isolés comme bruit.
- **Mélanges gaussiens** : produisent des probabilités d'appartenance et permettent des groupes ellipsoïdaux.

Le choix dépend de la géométrie attendue, du bruit, de la dimension et du type de résultat recherché.

Analyse en Composantes Principales (ACP)

L'Analyse en Composantes Principales (ACP) est une technique statistique de réduction de dimensionnalité. Elle transforme les données en un nouveau système de coordonnées où la plus grande variance est capturée sur les premiers axes, appelés composantes principales.



Formulation mathématique de l'ACP

Soit X une matrice de données de dimension $n \times p$ (n observations, p variables), centrée (moyenne nulle). L'ACP cherche à trouver les vecteurs propres et les valeurs propres de la matrice de covariance $C = \frac{1}{n-1} X^T X$.

Les composantes principales sont données par les vecteurs propres u_k de C, ordonnés par leurs valeurs propres correspondantes λ_k en ordre décroissant.

La k-ième composante principale de l'ensemble de données est donnée par :

$$Z_k = X u_k$$

Les valeurs propres λ_k représentent la variance expliquée par chaque composante principale.

Décomposition en valeurs propres

La matrice de covariance C peut être décomposée comme suit :

$$C = VLV^T$$

où $V = [u_1, u_2, \ldots, u_p]$ est la matrice des vecteurs propres et L est une matrice diagonale des valeurs propres λ_k.

La contribution de chaque composante principale à la variance totale est donnée par :

$$\frac{\lambda_k}{\sum_{i=1}^{p} \lambda_i}$$

Interprétation des composantes principales

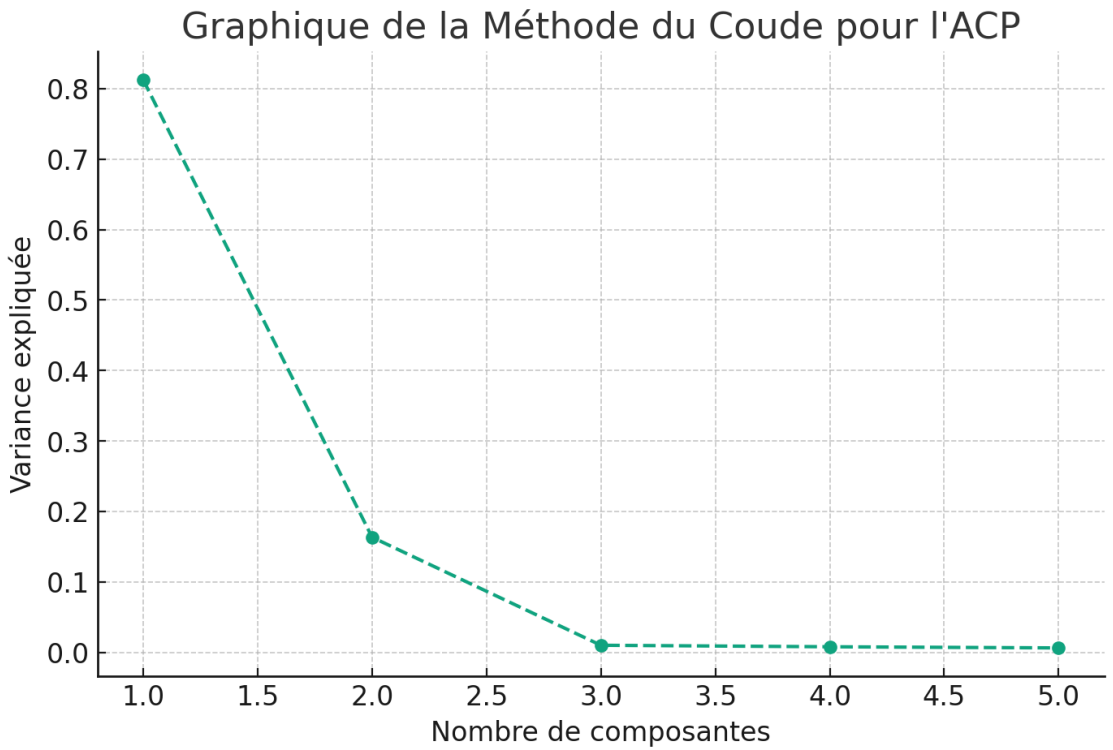
Chaque composante principale est une combinaison linéaire des variables d'origine. Les coordonnées des vecteurs propres décrivent les directions ; les *loadings*, selon la convention retenue, incorporent aussi l'échelle donnée par les valeurs propres.

- Un coefficient absolu élevé indique que la variable intervient fortement dans la direction considérée, sous réserve de la mise à l'échelle initiale.
- La direction et la magnitude des composantes principales peuvent être visualisées sur un biplot.

Choix du nombre de composantes principales

Le nombre de composantes peut être choisi à partir de la variance expliquée, de l'erreur de reconstruction ou des performances de la tâche menée ensuite.

Le *scree plot* représente les valeurs propres, ou la variance expliquée par composante, selon leur rang. La recherche d'un coude reste une règle heuristique.



Réduction de dimensionnalité

L'ACP est souvent utilisée pour réduire la dimensionnalité des données en conservant les composantes qui capturent la majorité de la variance.

- Cela permet une visualisation simplifiée des données en 2D ou 3D.
- La réduction de dimensionnalité peut également aider à améliorer l'efficacité des algorithmes d'apprentissage supervisé.

L'ACP a de nombreuses applications dans différents domaines :

- Analyse de données en biologie, finance, marketing.
- Traitement d'images et de signaux.
- Reconnaissance de motifs.

Limitations de l'ACP

Bien que l'ACP soit un outil puissant, elle présente certaines limitations :

- Sensibilité aux outliers.
- Sensibilité à la mise à l'échelle des variables et instabilité des directions lorsque plusieurs valeurs propres sont très proches.
- La réduction de dimensionnalité peut entraîner une perte d'information importante.

Principe de l'Isolation Forest

L'Isolation Forest est une technique de détection d'anomalies basée sur l'isolement des observations. Son efficacité repose sur l'hypothèse que les anomalies sont "faciles à isoler" par rapport aux observations normales.

- Fonctionne en construisant des arbres d'isolement à partir de sous-ensembles de données.
- Isoler une observation signifie la séparer des autres par des divisions aléatoires de l'espace des caractéristiques.
- Moins de divisions sont nécessaires pour isoler une anomalie, ce qui constitue le fondement du score d'anomalie.

Construction des arbres d'isolement

Les arbres d'isolement sont construits de manière récursive :

1. Sélection aléatoire d'un sous-ensemble de données.
2. Choix aléatoire d'une caractéristique et d'une valeur de seuil pour diviser le sous-ensemble.
3. Répétition des divisions jusqu'à l'isolement des observations ou atteinte d'une limite de profondeur prédéfinie.

Chaque arbre est ainsi unique, offrant une perspective différente sur les données.

Calcul du score d'anomalie

Le score d'anomalie reflète la facilité avec laquelle une observation est isolée :

- Basé sur la longueur du chemin moyen (nombre de divisions) pour isoler une observation dans les arbres d'isolement.
- Un chemin court conduit à un score d'anomalie élevé ; ce score n'est pas une probabilité calibrée.
- Le score est normalisé pour se situer entre 0 et 1, où les valeurs proches de 1 indiquent des anomalies.

Formule du score d'anomalie :

$$s(x, \psi) = 2^{-\frac{\mathbb{E}[h(x)]}{c(\psi)}}$$

où ψ est la taille des sous-échantillons, $\mathbb{E}[h(x)]$ la longueur moyenne du chemin et $c(\psi)$ la longueur moyenne attendue dans un arbre de recherche binaire.

Avantages et limitations de l'Isolation Forest

Avantages :

- Coût typique $O(t\,\psi \log \psi)$ pour t arbres construits sur des sous-échantillons de taille ψ.
- Nombre limité de paramètres principaux : taille des sous-échantillons, nombre d'arbres et contamination éventuelle.
- Ne nécessite ni distance globale ni estimation explicite de densité.

Limitations :

- La performance peut varier en fonction du choix de la taille de l'échantillon et du nombre d'arbres.
- Peut être moins efficace pour identifier des groupes d'anomalies denses.

Applications de l'Isolation Forest

L'Isolation Forest est largement utilisé dans des domaines variés pour sa capacité à identifier rapidement les anomalies :

- Détection de fraude financière.
- Maintenance prédictive dans l'industrie.
- Sécurité informatique et détection d'intrusions.
- Analyse de données médicales pour identifier les cas atypiques.

Sa flexibilité et sa simplicité en font un outil largement utilisé par les professionnels de la data science.

Prétraitement des données

Prétraitement des données

Le prétraitement des données est une étape cruciale en machine learning qui peut avoir des conséquences majeures sur les performances des modèles et leur interprétation. Le prétraitement vise à rendre les données plus appropriées pour le modèle. Cela inclut un certain nombre d'opérations telles que :

- Nettoyage des données.
- Gestion des valeurs manquantes.
- Normalisation et standardisation.
- Encodage des variables catégorielles.
- Prétraitements spécifiques à certaines données (texte, images, séries temporelles, etc.).

Nettoyage des données

Le nettoyage des données implique de détecter et de corriger, ou de supprimer, les erreurs et les incohérences. Cela inclut :

- Correction des erreurs de saisie.
- Identification et traitement des valeurs aberrantes.
- Suppression des doublons.
- Harmonisation des formats, unités et catégories.

Normalisation et standardisation

La mise à l'échelle est importante pour les méthodes fondées sur les distances, les produits scalaires ou les pénalités.

- **Normalisation :** elle redimensionne les valeurs dans un intervalle qui est généralement [0, 1].

$$X_{\text{normalisé}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

- **Standardisation :** elle redimensionne les données pour qu'elles aient une moyenne de 0 et un écart type de 1.

$$X_{\text{standardisé}} = \frac{X - \mu}{\sigma}$$

où μ est la moyenne et σ est l'écart-type de la caractéristique X.

Méthodes particulièrement concernées :

- Machines à vecteurs de support (SVM).
- Régressions Ridge et Lasso.
- Régression logistique ; OLS pour comparer les coefficients.
- Analyse en composantes principales.
- K-means.

Prétraitement sans fuite

Les paramètres du prétraitement doivent être appris uniquement sur le jeu d'entraînement.

- Moyennes, écarts-types, catégories, valeurs d'imputation et composantes principales sont estimés sur l'entraînement.
- Les mêmes transformations sont ensuite appliquées aux jeux de validation et de test.
- En validation croisée, le prétraitement est réajusté dans chaque pli.
- Une chaîne de traitement garantit que cette règle est respectée lors de l'entraînement et du déploiement.

Encodage des variables catégorielles

Les modèles de machine learning travaillent sur des représentations numériques. Deux encodages de base sont :

- **Encodage One-Hot** crée une nouvelle colonne pour chaque catégorie.
- **Encodage ordinal** attribue un rang numérique uniquement lorsque les catégories possèdent un ordre réel.

Pour les variables à forte cardinalité, on peut utiliser le hachage, des encodages régularisés par la cible ou des représentations apprises. Un encodage utilisant la cible doit être calculé hors pli pour les observations d'entraînement et sans exploiter les jeux de validation ou de test.

Gestion des valeurs manquantes

Les valeurs manquantes représentent l'absence d'information dans un ensemble de données et constituent un défi courant en machine learning et analyse de données. Elles peuvent survenir pour diverses raisons, telles que :

- Erreurs de saisie des données.
- Perte de données lors de la transmission.
- Non-réponse dans les enquêtes ou questionnaires.
- Suppression intentionnelle de données pour des raisons de confidentialité.

La gestion appropriée des valeurs manquantes est cruciale pour maintenir la qualité et la fiabilité des modèles prédictifs. Elle implique des techniques telles que l'imputation, la suppression des observations manquantes, ou l'utilisation de modèles capables de gérer directement les données manquantes.

Typologies des données manquantes

Comprendre la nature des données manquantes est crucial pour choisir la méthode de traitement appropriée.

- **MCAR (Missing Completely At Random)** : La probabilité qu'une donnée soit manquante est la même pour toutes les observations. L'absence de données est totalement indépendante des données observées ou manquantes.
- **MAR (Missing At Random)** : La probabilité qu'une donnée soit manquante dépend des données observées et non des données manquantes elles-mêmes.
- **MNAR (Missing Not At Random)** : La probabilité qu'une donnée soit manquante dépend des données manquantes elles-mêmes.

La distinction entre ces catégories influence la stratégie d'imputation et l'analyse des données.

Stratégies de base pour les données manquantes

Selon la typologie, différentes stratégies peuvent être adoptées. Mais, avant toute action, il faut d'abord essayer de comprendre pourquoi il y a des données manquantes, car le fait même qu'il y en ait constitue une information en soi.

Il existe différentes stratégies pour traiter les données manquantes :

- Suppression de lignes ou de colonnes
- Imputation simple (moyenne, médiane, mode, etc.)
- Techniques d'imputation prédictive pour estimer les valeurs manquantes.
- Modélisation explicite du mécanisme de données manquantes.

L'identification du type de données manquantes aide à choisir la méthode la plus adaptée et à minimiser le biais introduit par l'imputation.

Imputation simple vs. Imputation multiple

- **Imputation simple** :
  - Remplace les valeurs manquantes par une estimation (moyenne, médiane, etc.).
  - Facile à implémenter mais ne tient pas compte de l'incertitude autour des valeurs imputées.

- **Imputation multiple** :
  - Génère plusieurs ensembles complets de données en remplaçant les valeurs manquantes par un ensemble de valeurs plausibles.
  - Propage l'incertitude des valeurs imputées lorsque le modèle d'imputation est correctement spécifié.

L'imputation multiple est principalement justifiée sous l'hypothèse MAR, avec un modèle d'imputation correctement spécifié. Sous MNAR, il faut modéliser le mécanisme de non-réponse et mener des analyses de sensibilité.

Techniques avancées d'imputation

Des techniques avancées peuvent mieux gérer la complexité des données manquantes :

- **Imputation KNN (k-Nearest Neighbors)** : Utilise les k observations les plus similaires pour imputer les valeurs manquantes.
- **Imputation par d'autres modèles prédictifs** : Utilise des modèles comme les arbres de décision, forêts aléatoires ou réseaux de neurones pour prédire les valeurs manquantes.
- **Imputation multiple bayésienne** : génère des valeurs à partir de distributions prédictives ; MCMC peut servir à échantillonner les paramètres et les valeurs manquantes.

Ces méthodes tentent de modéliser la structure sous-jacente des données pour une imputation précise et pour minimiser le biais.

Considérations finales sur le traitement des données manquantes

- Identifier la typologie des données manquantes est essentiel pour choisir la méthode d'imputation appropriée.
- Les méthodes d'imputation doivent être choisies en tenant compte de l'impact sur la distribution des données et sur les analyses ultérieures.
- L'imputation multiple propage l'incertitude d'imputation lorsque son modèle et les hypothèses sur le mécanisme de non-réponse sont adaptés.
- L'exploration des données et la compréhension du contexte sont cruciales pour interpréter correctement les raisons derrière les données manquantes et pour choisir la méthode d'imputation la plus adéquate.
- Il est important de documenter le processus d'imputation et d'évaluer l'impact de différentes stratégies sur les résultats du modèle.

Considérer le traitement des données manquantes comme une composante intégrale de la préparation des données peut améliorer significativement la qualité et la fiabilité des analyses de machine learning.

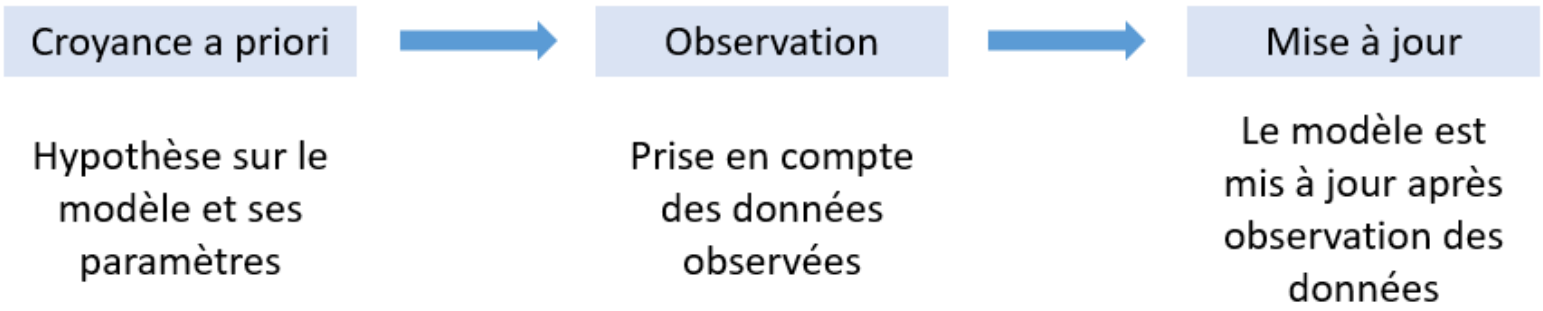
Approche bayésienne

Introduction aux méthodes bayésiennes

Les méthodes bayésiennes fournissent un cadre pour construire des modèles prédictifs et quantifier l'incertitude sur leurs paramètres. Le théorème de Bayes met à jour une distribution a priori à partir des données observées.

L'approche bayésienne repose sur deux concepts :

1. **La probabilité a priori** : La connaissance ou l'hypothèse initiale sur les paramètres avant d'observer les données. Elle reflète les croyances préalables sur les paramètres du modèle.
2. **La distribution a posteriori** : distribution des paramètres après prise en compte des données.



Formulation mathématique de l'approche bayésienne

Concrètement, la probabilité du modèle est mise à jour en utilisant le théorème de Bayes :

$$P(\theta|\text{données}) = \frac{P(\text{données}|\theta) \cdot P(\theta)}{P(\text{données})}$$

où :

- $P(\theta|\text{données})$ désigne la densité a posteriori des paramètres θ.
- $P(\text{données}|\theta)$ est la vraisemblance des données observées sous l'hypothèse des paramètres θ.
- $P(\theta)$ désigne la distribution a priori des paramètres, ou sa densité dans le cas continu.
- $P(\text{données})$ est la probabilité marginale des données, agissant comme une constante de normalisation.

Dans les modèles conjugués, la normalisation peut être analytique. Dans les modèles complexes, elle peut être coûteuse, mais elle s'élimine souvent dans les rapports utilisés par MCMC.

Estimation des paramètres dans les modèles bayésiens

L'inférence bayésienne consiste à déterminer ou approximer la distribution a posteriori des paramètres :

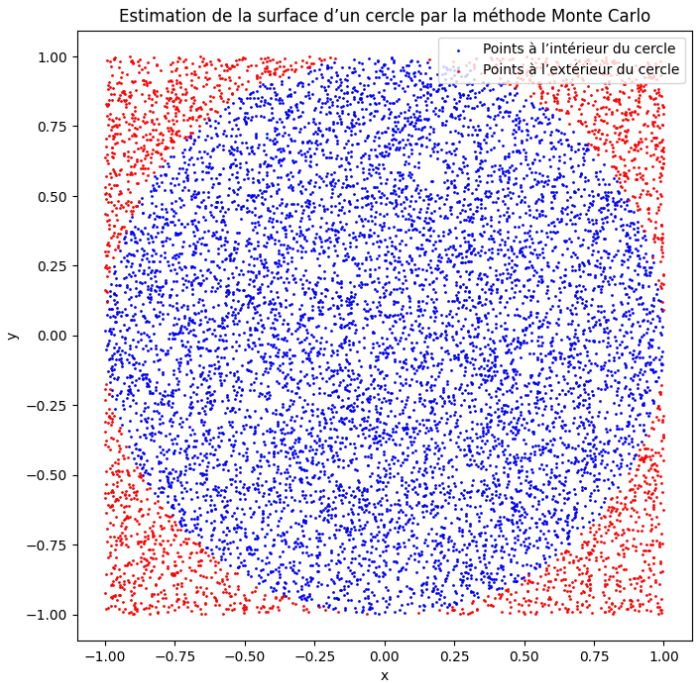
$$p(\boldsymbol{\theta} \mid D) \propto p(D \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}).$$

L'estimateur MAP, $\widehat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid D)$, n'est qu'un résumé ponctuel de cette distribution.
Les méthodes d'estimation des paramètres incluent :

1. **Méthodes analytiques** : Dans certains cas, la distribution a posteriori peut être calculée directement à partir de formules analytiques, souvent lorsque le modèle prior et la vraisemblance sont conjugués.
2. **Méthodes d'échantillonnage** : Lorsque les solutions analytiques ne sont pas réalisables, des techniques d'échantillonnage telles que la chaîne de Markov Monte Carlo (MCMC) sont utilisées pour générer des échantillons de la distribution a posteriori.
3. **Approximation variationnelle** : Une alternative aux méthodes d'échantillonnage, où l'objectif est de trouver une distribution simple qui se rapproche au mieux de la distribution a posteriori complexe.

Monte-Carlo

Le graphique illustre une estimation de π par Monte-Carlo. On génère des points uniformément dans le carré $[-1, 1] \times [-1, 1]$ et on mesure la proportion située dans le disque unité.
Si N_{int} points sur N sont dans le disque, alors $\widehat{\pi} = 4N_{\text{int}}/N$. Cette méthode utilise des tirages indépendants ; MCMC est utile lorsque l'on ne sait pas tirer directement dans la distribution cible.



Principe des techniques MCMC

MCMC construit une chaîne de Markov qui a pour distribution stationnaire la distribution cible, souvent la distribution postérieure en inférence bayésienne. Les

techniques MCMC s'articulent autour des étapes suivantes :

- Partir d'un état initial $\theta^{(0)}$.
- Proposer un nouvel état θ' à partir de l'état actuel $\theta^{(n)}$ en utilisant une distribution de proposition $Q(\theta'|\theta^{(n)})$.
- Accepter θ' avec une probabilité basée sur le ratio d'acceptation pour passer à $\theta^{(n+1)}$, sinon répéter $\theta^{(n)}$.

Algorithme de Metropolis-Hastings

Metropolis-Hastings est un algorithme MCMC très populaire qui permet d'échantillonner à partir de la distribution postérieure $P(\theta|D)$.

1. Générer un état candidat θ' en utilisant $Q(\theta'|\theta^{(n)})$.
2. Calculer le ratio d'acceptation a:

$$a(\theta', \theta^{(n)}) = \min\left(1, \frac{P(D|\theta')P(\theta')Q(\theta^{(n)}|\theta')}{P(D|\theta^{(n)})P(\theta^{(n)})Q(\theta'|\theta^{(n)})}\right)$$

3. Accepter θ' avec la probabilité a, sinon garder $\theta^{(n)}$ comme $\theta^{(n+1)}$.

Sous des conditions de régularité, la chaîne admet la postérieure comme distribution stationnaire et converge vers elle.

Diagnostic des chaînes MCMC

Une simulation MCMC doit être contrôlée avant d'utiliser les échantillons :

- comparer plusieurs chaînes initialisées différemment ;
- inspecter les trajectoires et l'autocorrélation ;
- calculer une taille effective d'échantillon ;
- vérifier un diagnostic inter-chaînes tel que $\widehat{R}$;
- distinguer la phase d'adaptation des échantillons utilisés pour l'inférence.

Régression bayésienne

La régression bayésienne est une extension des méthodes de régression classiques qui incorpore l'incertitude dans les estimations des paramètres de régression. En utilisant l'approche bayésienne, chaque paramètre de régression est traité comme une variable aléatoire avec sa propre distribution a priori, qui est mise à jour en une distribution a posteriori en tenant compte des données observées.

La formulation de base de la régression bayésienne linéaire est :

$$y = \mathbf{X}\boldsymbol{\beta} + \epsilon$$

où y est le vecteur de réponse, $\mathbf{X}$ est la matrice de conception, $\boldsymbol{\beta}$ est le vecteur des coefficients de régression, et ϵ est le terme d'erreur, souvent supposé suivre une distribution normale avec une moyenne de zéro.

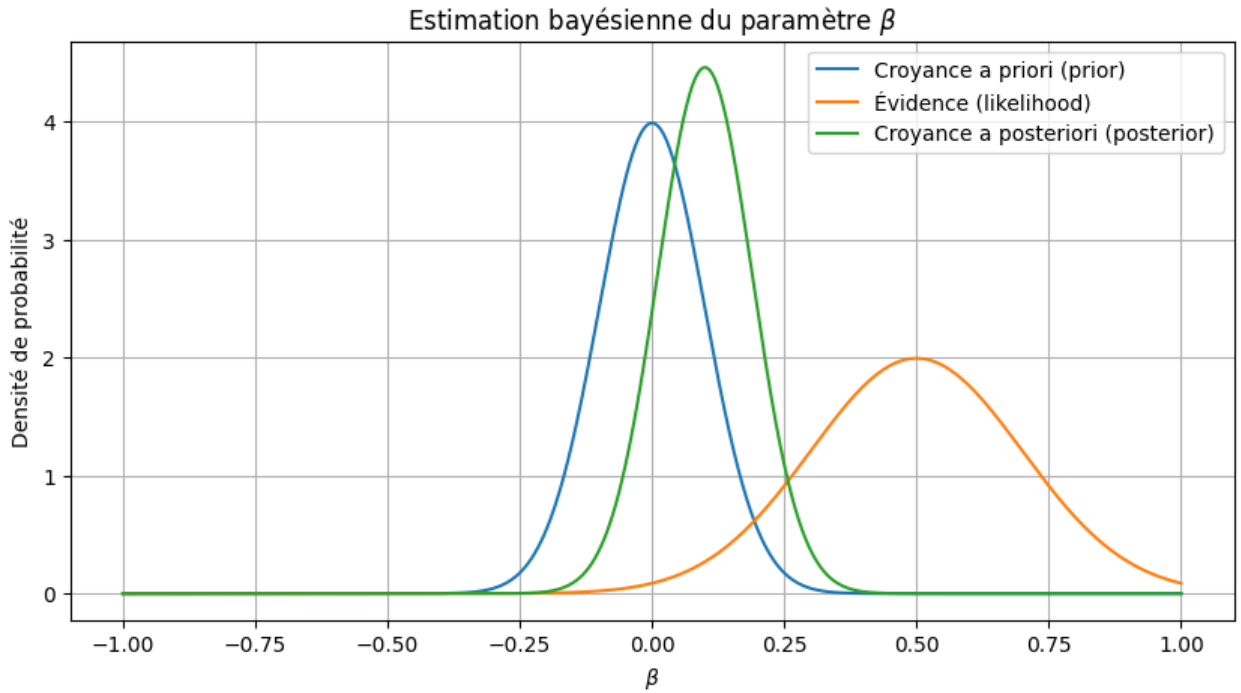
Le processus d'estimation bayésienne se concentre sur la détermination de la distribution a posteriori des coefficients $\boldsymbol{\beta}$:

$$P(\boldsymbol{\beta}|y, \mathbf{X}) \propto P(y|\boldsymbol{\beta}, \mathbf{X}) \cdot P(\boldsymbol{\beta})$$

Avantages de la régression bayésienne

La régression bayésienne permet de :

- Incorporer des connaissances a priori dans les estimations des coefficients.
- Obtenir une mesure de l'incertitude pour chaque estimation de coefficient à travers la distribution a posteriori.
- Faire des prédictions et calculer des intervalles de prédiction qui prennent en compte l'incertitude de tous les paramètres du modèle.



Classification bayésienne

La classification bayésienne utilise la probabilité conditionnelle et le théorème de Bayes pour calculer la probabilité de chaque classe.

La formule fondamentale de la classification bayésienne est le calcul de la probabilité a posteriori pour chaque classe C_k donnée une observation $\mathbf{x}$:

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k) \cdot P(C_k)}{P(\mathbf{x})}$$

où :

- $P(C_k|\mathbf{x})$ est la probabilité a posteriori de la classe C_k étant donné l'observation $\mathbf{x}$.
- $P(\mathbf{x}|C_k)$ est la vraisemblance de l'observation $\mathbf{x}$ dans la classe C_k.
- $P(C_k)$ est la probabilité a priori de la classe C_k.
- $P(\mathbf{x})$ est la probabilité marginale de l'observation $\mathbf{x}$, souvent calculée comme la somme des vraisemblances de $\mathbf{x}$ sur toutes les classes pondérée par leur probabilité a priori.

Classification avec Naive Bayes

Le classificateur Naive Bayes est un modèle probabiliste basé sur le théorème de Bayes, avec une hypothèse simplificatrice d'indépendance conditionnelle entre les caractéristiques données la classe.

Points clés :

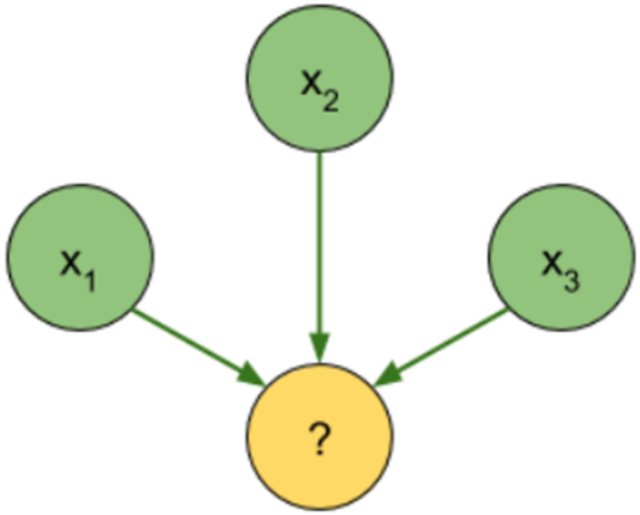
- Facile à construire et particulièrement utile pour de très grands ensembles de données.
- Malgré son hypothèse naïve d'indépendance, il peut être étonnamment efficace.
- Largement appliqué à la classification de documents et au filtrage anti-spam.

Hypothèse d'indépendance dans Naive Bayes

L'hypothèse de Naive Bayes suppose que les caractéristiques sont mutuellement indépendantes conditionnellement à la classe.

Cela simplifie le calcul de $P(x|C_k)$ car, pour p caractéristiques :

$$P(x_1, \ldots, x_p|C_k) = \prod_{j=1}^{p} P(x_j|C_k).$$



Avantages et limites de Naive Bayes

Avantages :

- Efficacité en temps et en espace, même avec de grands ensembles de données.
- Souvent efficace sur les représentations creuses de grande dimension, notamment pour le texte.
- Prédiction probabiliste rapide lorsque les distributions conditionnelles sont simples à estimer.

Limites :

- L'hypothèse d'indépendance peut être irréaliste et peut affecter la performance.
- Les probabilités nulles dues à des catégories absentes nécessitent un lissage, par exemple le lissage de Laplace.
- Les probabilités prédites peuvent être mal calibrées lorsque les dépendances entre caractéristiques sont fortes.

Deep learning

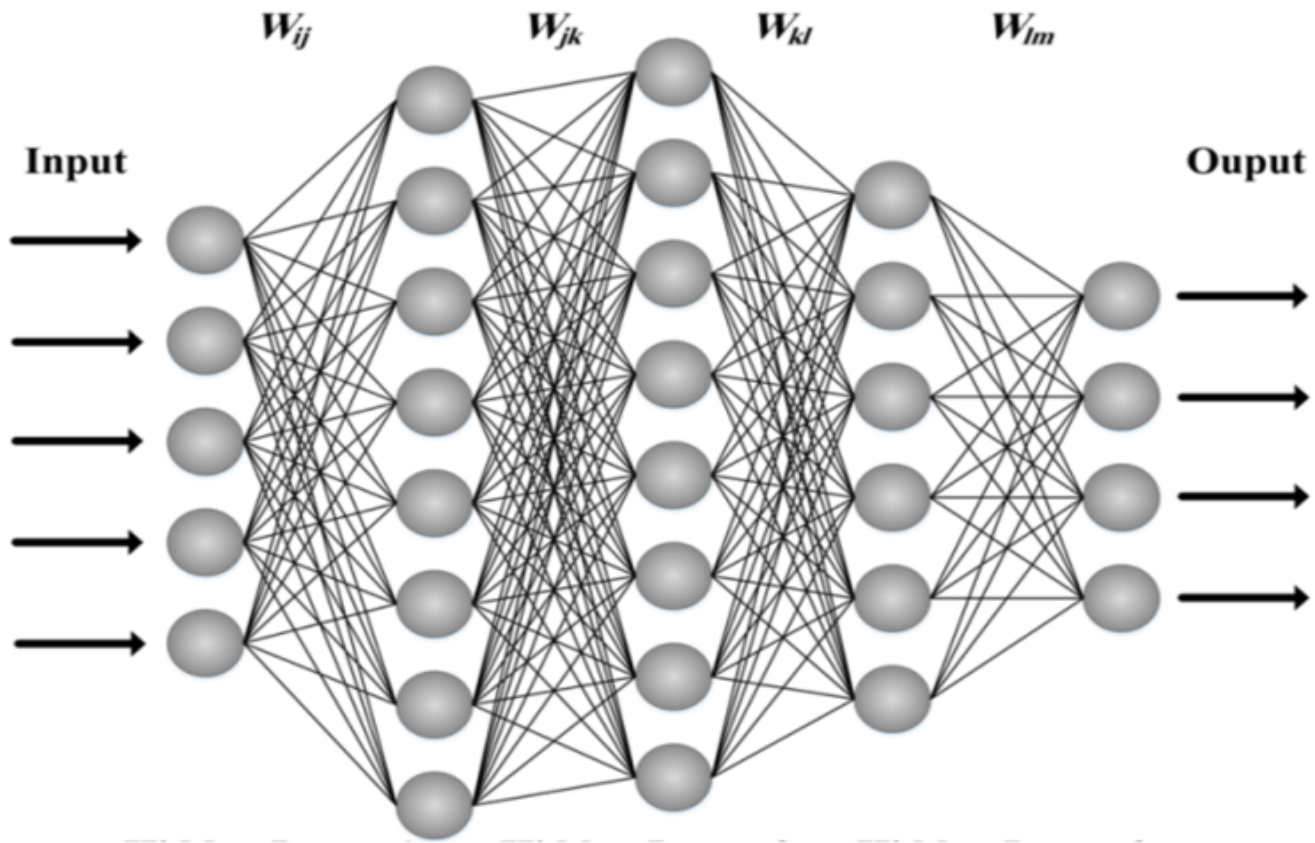
Introduction aux réseaux de neurones

Définition : un réseau de neurones est une composition paramétrée de transformations affines et de fonctions non linéaires. Ses paramètres sont ajustés à partir des données pour approximer une fonction.

Caractéristiques :

- **Apprentissage de représentations** : extraction de caractéristiques adaptées à la tâche à partir des données.
- **Modélisation non linéaire** : aptitude à représenter des relations complexes dans les données.
- **Données de grande dimension** : les architectures spécialisées exploitent la structure des images, du texte, des signaux ou des graphes.
- **Flexibilité** : application à la classification, la régression, la génération et l'apprentissage de représentations.

Architecture d'un réseau multicouche



Le neurone artificiel

Modèle mathématique : chaque neurone effectue une somme pondérée de ses entrées, ajoute un biais puis applique une fonction d'activation.

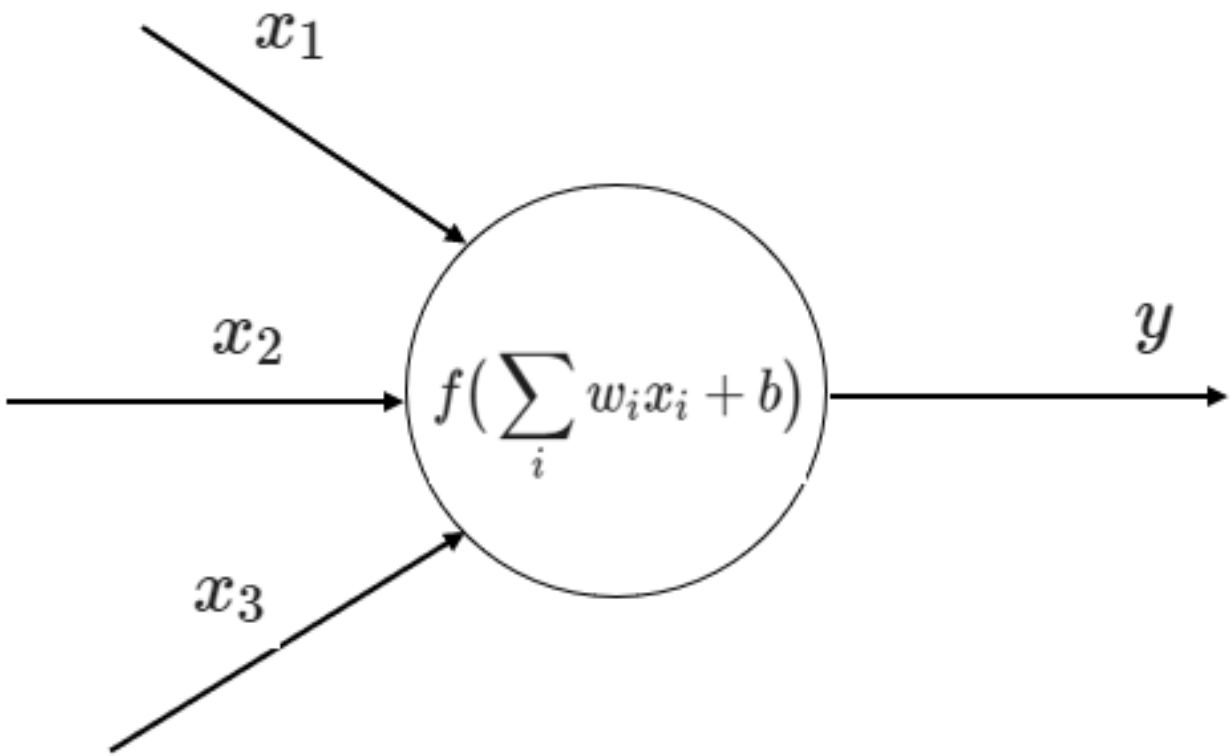
$$z_i = \sum_{j=1}^{m} w_{ij} x_j + b_i$$

$$a_i = f(z_i)$$

où :

- x_j représente l'entrée du neurone,
- w_{ij} est le poids associé à l'entrée x_j,
- b_i est le biais du neurone,
- f est la fonction d'activation,
- z_i est la préactivation,
- a_i est la sortie activée du neurone.

Calcul dans un neurone



Fonctions d'activation

Les fonctions d'activation permettent aux modèles d'apprendre des relations non linéaires.

- **Sigmoïde** : $\sigma(z) = \frac{1}{1+e^{-z}}$, plage de sortie $(0, 1)$, utilisée pour paramétrer la probabilité de sortie en classification binaire.
- **Tangente hyperbolique (tanh)** : $\tanh(z)$, plage de sortie $(-1, 1)$ et sortie centrée en zéro.
- **ReLU** : $f(z) = \max(0, z)$, non saturante pour $z > 0$; elle atténue certains gradients évanescents mais peut produire des unités inactives.
- **Leaky ReLU** : $f(z) = \max(\alpha z, z)$, variante de ReLU qui permet un petit gradient lorsque $z < 0$.

Softmax et entropie croisée

Pour une classification à K classes, la couche de sortie produit des logits $z_1, \ldots, z_K$. La fonction softmax les transforme en probabilités :

$$p_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, \qquad \sum_{k=1}^{K} p_k = 1.$$

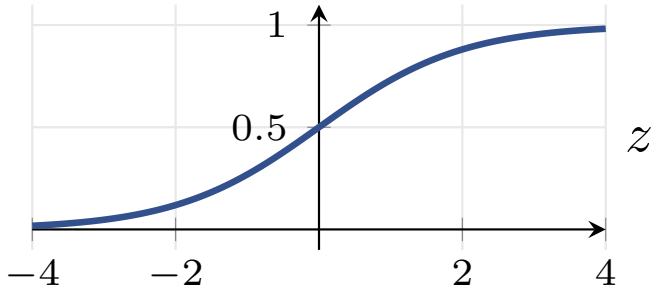
Pour une cible codée par $y_k \in \{0, 1\}$, l'entropie croisée multiclasse est

$$L(y, p) = -\sum_{k=1}^{K} y_k \log p_k.$$

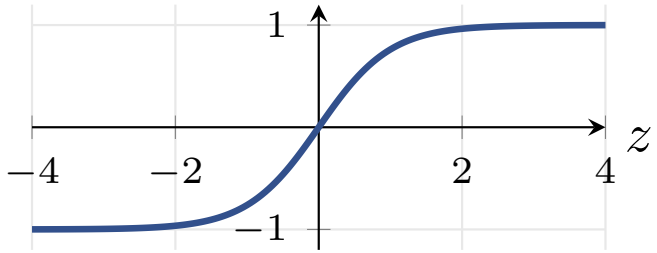
- Minimiser cette perte revient à maximiser la vraisemblance du modèle catégoriel.
- En calcul numérique, on stabilise softmax en soustrayant $\max_j z_j$ à tous les logits.
- La classe prédite est $\arg\max_k p_k$; les probabilités peuvent néanmoins nécessiter une calibration.

Courbes des fonctions d'activation

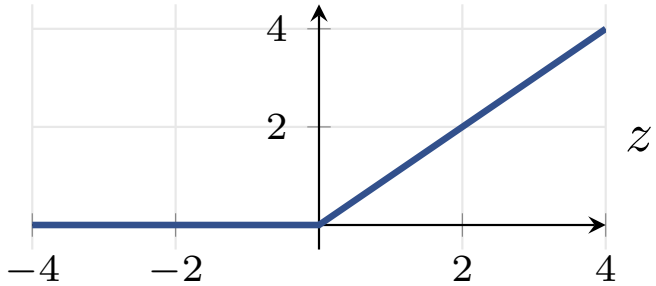
Sigmoïde



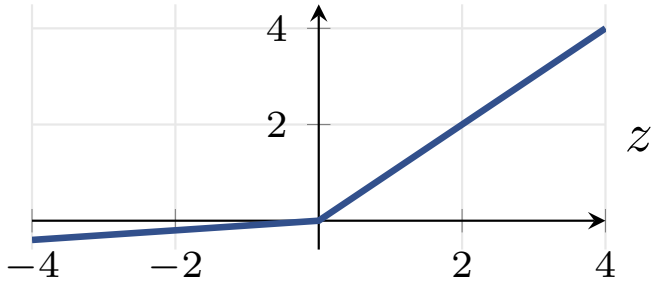
Tangente hyperbolique



ReLU



Leaky ReLU ($\alpha = 0{,}1$)



Descente de gradient stochastique

Principe : l'entraînement ajuste les poids afin de minimiser une fonction de perte sur des mini-lots.

Descente de gradient stochastique (SGD) :

- Méthode d'optimisation utilisée pour mettre à jour les poids du réseau de manière itérative.
- À chaque itération, un sous-ensemble (batch) de données est utilisé pour calculer le gradient de la fonction de coût.
- Les poids sont mis à jour dans la direction opposée du gradient pour réduire l'erreur.

$$w_{new} = w_{old} - \eta \cdot \nabla_w J(w)$$

où :

- w_{old} et w_{new} sont les valeurs des poids avant et après la mise à jour,
- η est le taux d'apprentissage,
- $\nabla_w J(w)$ est le gradient de la fonction de coût par rapport aux poids.

Optimisation des réseaux

La qualité de l'entraînement dépend de plusieurs choix couplés :

- initialisation des poids adaptée aux fonctions d'activation ;
- taux d'apprentissage et calendrier de décroissance ;
- taille des mini-lots ;
- optimiseur : SGD avec momentum, RMSProp ou Adam ;
- normalisation des activations et contrôle des gradients.

Ces choix sont évalués sur le jeu de validation, avec un protocole reproductible.

Stabiliser l'entraînement

Initialisation des poids

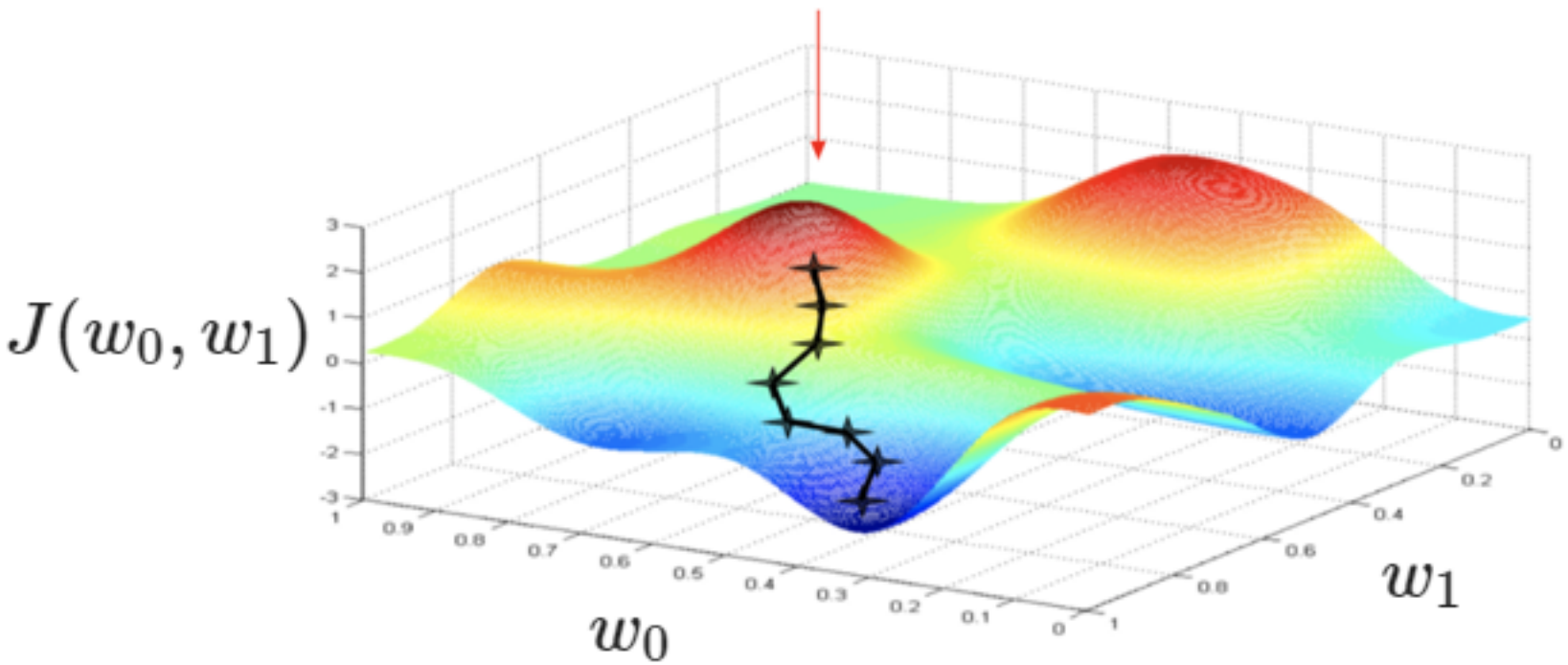
- Xavier : variance approximative $2/(n_{\text{in}} + n_{\text{out}})$, adaptée notamment à tanh.
- He : variance approximative $2/n_{\text{in}}$, adaptée aux activations de type ReLU.
- Ces choix limitent l'explosion ou l'évanouissement des activations et des gradients au début de l'entraînement.

Normalisation par lots

$$\widehat{z} = \frac{z - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \qquad \text{BN}(z) = \gamma \widehat{z} + \beta.$$

- Les statistiques (μ_B, σ_B^2) sont calculées sur le mini-lot pendant l'entraînement.
- En inférence, des statistiques agrégées pendant l'entraînement sont utilisées.

Trajectoire de la descente de gradient

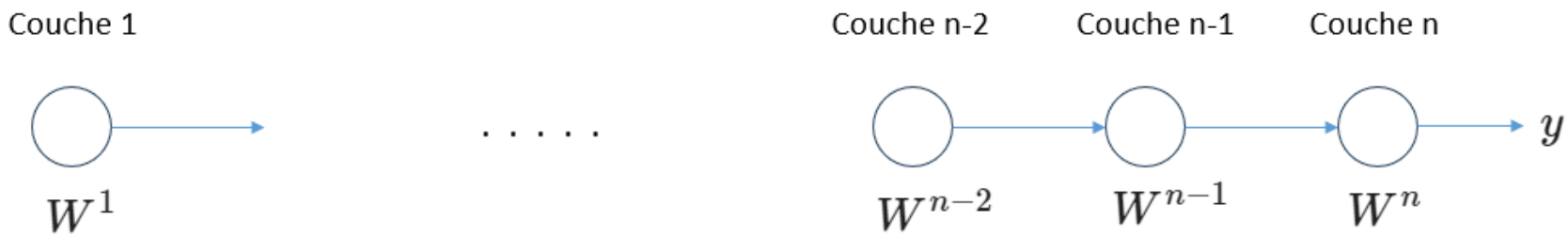


Calcul des gradients dans un réseau profond

Dans les réseaux de neurones profonds, l'erreur calculée à la sortie du réseau dépend indirectement des poids des couches cachées. Ce lien indirect rend difficile de savoir comment ajuster ces poids pour réduire l'erreur.

Implications pour l'entraînement :

- **Propagation de l'erreur** : Sans un mécanisme pour propager l'erreur de la sortie vers les couches antérieures, il est impossible de déterminer l'impact de chaque poids sur l'erreur finale.
- **Complexité de l'ajustement des poids** : Chaque poids dans les couches cachées affecte l'erreur de sortie de manière complexe, nécessitant une méthode précise pour leur ajustement.



Rétropropagation du gradient

Pour la couche ℓ :

$$z^{(\ell)} = W^{(\ell)} a^{(\ell-1)} + b^{(\ell)}, \qquad a^{(\ell)} = \phi^{(\ell)}(z^{(\ell)}).$$

En notant $\delta^{(\ell)} = \partial J / \partial z^{(\ell)}$, la règle de chaîne donne :

$$\delta^{(\ell)} = \left(W^{(\ell+1)}\right)^T \delta^{(\ell+1)} \odot \phi^{(\ell)\prime}(z^{(\ell)}).$$

Les gradients des paramètres sont :

$$\frac{\partial J}{\partial W^{(\ell)}} = \delta^{(\ell)} \left(a^{(\ell-1)}\right)^T, \qquad \frac{\partial J}{\partial b^{(\ell)}} = \delta^{(\ell)}.$$

La rétropropagation réutilise les gradients calculés de la sortie vers l'entrée.

Régularisation et arrêt anticipé

Un réseau trop flexible ou entraîné trop longtemps peut continuer à réduire l'erreur d'apprentissage alors que l'erreur de validation augmente.

- **Pénalités sur les poids** : ajout d'un terme L_1 ou L_2 à la fonction de perte.
- **Dropout** : masque aléatoire d'activations pendant l'entraînement ; il est désactivé en inférence.
- **Augmentation de données** : transformations cohérentes avec les invariances attendues du problème.
- **Arrêt anticipé** : conserver les paramètres correspondant au meilleur score de validation et arrêter après un nombre fixé d'époques sans amélioration.

Le nombre maximal d'époques, la patience et la métrique suivie font partie du protocole de réglage. Le jeu de test n'intervient pas dans cette décision.

Bibliographie 1/3

- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2e éd.). Springer.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning* (2e éd.). Springer.
- Bishop, C. M., & Bishop, H. (2024). *Deep Learning: Foundations and Concepts*. Springer.
- Murphy, K. P. (2022–2023). *Probabilistic Machine Learning*. MIT Press.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Azencott, C.-A. (2019). *Introduction au machine learning*. Dunod.

Bibliographie 2/3

- Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273–297.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32.
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29, 1189–1232.
- Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. *Proceedings of ICDM*, 413–422.
- Little, R. J. A., & Rubin, D. B. (2019). *Statistical Analysis with Missing Data* (3e éd.). Wiley.
- Gelman, A. et al. (2013). *Bayesian Data Analysis* (3e éd.). CRC Press.
- Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *JMLR*, 13, 281–305.

Bibliographie 3/3

- Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. *NeurIPS*.
- Cawley, G. C., & Talbot, N. L. C. (2010). On over-fitting in model selection and subsequent selection bias. *JMLR*, 11, 2079–2107.
- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *KDD*.
- Ke, G. et al. (2017). LightGBM: A highly efficient gradient boosting decision tree. *NeurIPS*.
- Prokhorenkova, L. et al. (2018). CatBoost: Unbiased boosting with categorical features. *NeurIPS*.
- Ioffe, S., & Szegedy, C. (2015). Batch normalization. *ICML*.
- Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *AISTATS*.
- He, K. et al. (2015). Delving deep into rectifiers. *ICCV*.

Merci de votre attention

redha.moulla@axia-conseil.com