MTH 8302 - Modèles de Régression et d'Analyse de Variance

Leçon 4 : Modèles Linéaires Généralisés et Méthodes Classiques d'Apprentissage Supervisé

Polytechnique Montréal - Hiver 2025

Chiheb Trabelsi

April 16, 2025

Table des Matières

1 Modèles Linéaires Généralisés

2 Regression Logistique

3 Analyse Linéaire Discriminante

4 Annexe

Modèles Linéaires Généralisés

Modèles Linéaires Généralisés : Introduction

Modèles Linéaires Généralisés : Introduction

- Dans le cadre de la régression linéaire, l'objectif est de modéliser une relation entre :
  - une variable réponse continue $Y \in \mathbb{R} \Rightarrow \mathbf{Y} \in \mathbb{R}^n$
  - à partir de variables explicatives $X_1, \ldots, X_p \in \mathbb{R}^n \Rightarrow \mathbf{X} \in \mathbb{R}^{n \times (p+1)}$.
- Cette relation est exprimée à l'aide d'un modèle de linéaire :

$$Y = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \text{où } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_n),$$

  - où $\boldsymbol{\beta} \in \mathbb{R}^{(p+1)}$ est le vecteur des paramètres à estimer.
- L'estimation des paramètres se fait typiquement par moindres carrés ou maximum de vraisemblance, ce qui revient à trouver l'estimateur $\hat{\boldsymbol{\beta}}$, tel que:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

- Cette approche est efficace **si les hypothèses sont vérifiées :**
  - Simple à mettre en oeuvre.
  - Interprétable puisque chaque β_j mesure l'effet linéaire de X_j.

Modèles Linéaires Généralisés : Introduction

- Cependant, les hypothèses considérées dans cette approche sont fortes :
  - $Y \in \mathbb{R}$ est nécessairement continue.
  - Les erreurs ϵ sont gaussiennes, centrées et homoscédastiques.
  - La relation entre Y et les X_j est linéaire.
- **Que se passe-t-il si Y n'est pas continue ?**
- **Exemples réalistes :**
  - $Y \in \{0, 1\}$ (binaire) : Prédire si un patient est malade (oui/non), si un client remboursera son prêt, etc.
  - $Y \in \mathbb{N}$: Nombre d'appels reçus dans un call center, nombre d'accidents dans une intersection, etc.
- **Problèmes de la régression linéaire dans ces cas :**
  - **Incohérence** : Prédictions négatives ou > 1.
  - **Statistiquement sous-optimale** : Ne tient pas compte de la distribution naturelle de Y.
  - **Peu performante** : Mauvaise généralisation.

Modèles Linéaires Généralisés : Introduction

- L'objectif est de généraliser les modèles linéaires classiques pour les rendre compatibles avec des situations où :
  - La variable cible Y n'est pas continue.
  - On veut modéliser des probabilités, des décomptes, voire même des proportions.
  - On souhaite garder une structure linéaire dans l'espace des variables explicatives X_j, tout en respectant la distribution naturelle de Y.
- **Solution :** Introduire les **Modèles Linéaires Généralisés** (GLM : *Generalized Linear Models*), qui permettent de :
  - Étendre la régression linéaire à **d'autres types de variables cibles.**
  - Garder une structure linéaire dans l'espace des prédicteurs X_j. Cela repose sur une idée simple mais efficace : Relier $\mathbb{E}[Y \mid X_1, \ldots, X_p]$, l'espérance de Y à une combinaison linéaire des variables explicatives X_j **via une fonction de lien adaptée notée g.**
  - **Tenir compte de la distribution de Y.** Cela permet d'intégrer des modèles comme :
    - La régression logistique (Y binaire $\Rightarrow Y \sim$ Bernoulli(p)).
    - La régression de Poisson (décompte $\Rightarrow Y \sim$ Poisson(λ)).
    - Etc.

La Famille Exponenielle des Distributions et Composantes des Modèles Linéaires Généralisés

La Famille Exponenielle et Composantes des GLM

- Notation adoptée :
  - $\mathbf{Y} \in \mathbb{R}^n$ et $Y \in \mathbb{R}$: variables aléatoires.
  - $\boldsymbol{y} \in \mathbb{R}^n$ et $y \in \mathbb{R}$: Réalisations (valeurs observées) respectives des variables aléatoire $\mathbf{Y}$ et Y

  Dans une fonction de densité, on écrit :

$$f_Y(y; \theta, \phi)$$

  - $f_Y(\cdot)$ désigne la densité de la variable aléatoire Y,
  - y est la valeur sur laquelle la densité est évaluée.

- L'expression de la densité de la famille exponentielle est donnée par :

$$f_Y(y; \theta, \phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right).$$

  - Y est bien la variable aléatoire (indiquée dans l'indice de f_Y),
  - y est la variable d'intégration ou d'évaluation (la réalisation de Y).

La Famille Exponenielle et Composantes des GLM

- L'expression de la densité de la famille exponentielle est donnée par :

$$f_Y(y;\theta,\phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right).$$

Signification des différentes composantes :

- y : réalisation de la variable aléatoire Y
- θ : paramètre naturel (ou canonique) de la distribution
- ϕ : paramètre de dispersion (ou fixé à 1 selon le contexte)
- $a(\phi)$: fonction de pondération de la dispersion (souvent $a(\phi) = \phi$)
- $b(\theta)$: fonction log-partition (fonction de normalisation logarithmique) qui garantit l'intégrabilité à 1. Elle détermine la moyenne et la variance de Y.
- $c(y,\phi)$: fonction indépendante de θ, absorbant les termes restants pour assurer la normalisation.

- **Propriétés des Moments de la distribution (Preuve Ici):**
  - $\mathbb{E}[Y] = \frac{\partial b(\theta)}{\partial \theta} = b'(\theta)$
  - $\text{Var}(Y) = a(\phi)\frac{\partial b(\theta)}{\partial^2 \theta} = a(\phi) \cdot b''(\theta)$
- **Remarque :** Cette forme permet d'exprimer une large famille de distributions usuelles dans un même cadre probabiliste.

La Famille Exponenielle et Composantes des GLM

Un GLM est défini par 3 composantes fondamentales :

1. **La distribution de la variable cible** Y
   - Y suit une distribution **appartenant à la famille exponentielle :**
     - **loi normale** (régression linéaire)
     - **loi de Bernoulli** (régression logistique)
     - **loi de Poisson** (décomptes)
     - autres : binomiale, gamma, inverse-gaussienne, etc.
   - Forme générale :

$$f_Y(y;\theta) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right)$$

2. **La fonction de lien** g
   - Relie $\boldsymbol{\mu} = \mathbb{E}[\mathbf{Y} \mid \mathbf{X}]$ au prédicteur linéaire $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$
   - $g(\boldsymbol{\mu}) = \boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$, (avec g appliqué élément par élément).
   - Le choix du lien dépend de la distribution de Y :
     - **Logit** pour Bernoulli
     - **Log** pour Poisson
     - **Identité** pour normale
3. **Le prédicteur linéaire** η
   - $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$.
   - Permet une estimation par maximum de vraisemblance.

La Famille Exponenielle et Composantes des GLM

2. La fonction de lien g

- Pour une $i$ème observation individuelle $\mathbf{x}_i$, la fonction de lien relie l'espérance conditionnelle $\mu_i = \mathbb{E}[Y_i \mid \mathbf{X}_i = \mathbf{x}_i]$ au prédicteur linéaire $\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$:

$$g(\mu_i) = \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$$

- Pour l'ensemble des n observations, on définit les vecteurs :

$$\boldsymbol{\mu} = \mathbb{E}[\mathbf{Y} \mid \mathbf{X}], \quad \boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$$

- La fonction de lien s'applique alors élément par élément :

$$g(\boldsymbol{\mu}) = \boldsymbol{\eta}$$

3. Le prédicteur linéaire $\boldsymbol{\eta}$

- Pour une observation : $\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$
- Pour l'échantillon : $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$
- Il constitue le coeur du modèle linéaire généralisé et permet de relier l'espace des variables explicatives à $\boldsymbol{\mu} = \mathbb{E}[\mathbf{Y} \mid \mathbf{X}]$.
- Il permet l'estimation de $\boldsymbol{\beta}$ par maximum de vraisemblance.

Étapes pour Vérifier qu'une loi appartient à la famille exp

- **Objectif :** Vérifier si la densité $f_Y(y)$ appartient à la famille exponentielle des distributions et pourrait donc s'écrire comme:

$$f_Y(y;\theta,\phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right)$$

- **Étapes à suivre :**
  1. **Écrire la densité dans sa forme classique.**
     Exemple : loi normale, Bernoulli, Poisson...
  2. **Mettre la densité sous forme exponentielle.**
     Identifier les termes permettant d'aboutir à la forme canonique.
  3. **Identifier les composantes du modèle exponentiel :**
     - θ : paramètre naturel
     - ϕ : paramètre de dispersion
     - $a(\phi)$: fonction de pondération
     - $b(\theta)$: fonction log-partition
     - $c(y,\phi)$: fonction de normalisation

Étapes pour Vérifier qu'une loi appartient à la famille exp

4. **Calculer les moments :**

$$\mathbb{E}[Y] = b'(\theta), \quad \text{Var}(Y) = a(\phi) \cdot b''(\theta)$$

5. **Déterminer la fonction de lien canonique :**

$$g(\mu) = \theta \quad \Rightarrow \quad \mu = b'(\theta)$$

6. **Exprimer le prédicteur linéaire du GLM :**

$$\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}, \quad \text{avec } g(\mu_i) = \eta_i$$

7. **Conclure :**
   - La loi appartient à la famille exponentielle.
   - Elle peut être modélisée via un GLM avec :
     - Fonction de lien : $g(\cdot)$

Exercice : Montrons que la loi normale de $Y \sim \mathcal{N}(\mu, \sigma^2)$ appartient à la famille exponentielle des densités.

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q1. Quelle est la densité d'une variable $Y \sim \mathcal{N}(\mu, \sigma^2)$?

A. $\exp\left(-\frac{(y-\mu)^2}{\sigma^2}\right)$

B. $\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$

C. $\frac{1}{2\pi\sigma^2} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q1. Quelle est la densité d'une variable $Y \sim \mathcal{N}(\mu, \sigma^2)$?
Réponse correcte : B
La densité d'une loi normale est bien :

$$f_Y(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q2. Développer l'expression $(y - \mu)^2$

A. $y^2 + \mu^2$

B. $y^2 - 2y\mu + -\mu^2$

C. $y^2 + 2y\mu + \mu^2$

D. $y^2 - 2\mu y + \mu^2$

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q2. Développer l'expression $(y-\mu)^2$

Réponse correcte : D

On a : $(y-\mu)^2 = y^2 - 2y\mu + \mu^2$

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q3. Mise sous forme exponentielle : quelle est la bonne forme ?

A. $\exp\left(-\frac{y^2}{2\sigma^2} + \frac{y\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} - \log(2\pi\sigma^2)\right)$

B. $\exp\left(\frac{y\mu - \mu^2/2}{\sigma^2} - \frac{y^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right)$

C. $\exp\left(\frac{(y-\mu)^2}{2\sigma^2}\right)$

D. $\exp\left(\frac{(y-\mu)^2}{\sigma^2}\right)$

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q3. Mise sous forme exponentielle : quelle est la bonne forme ?

Réponse correcte : B

C'est la bonne mise sous la forme : $\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)$

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q4. Quelles sont les expressions correctes ?

A. $\theta = \mu$, $a(\phi) = \phi = \sigma^2$, $b(\theta) = \frac{1}{2}\theta^2$

B. $\theta = y$, $b(\theta) = \mu^2$, $a(\phi) = \sigma^2$

C. $\theta = \sigma^2$, $b(\theta) = \theta^2$, $a(\phi) = \mu$

D. $\theta = \mu$, $b(\theta) = \mu^2$, $a(\phi) = \phi^2$

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q4. Quelles sont les expressions correctes ?

Réponse correcte : A

On identifie bien : $\theta = \mu$, $b(\theta) = \frac{1}{2}\theta^2$, $a(\phi) = \phi = \sigma^2$

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q5. Que valent les moments de Y ?

A. $\mathbb{E}[Y] = \theta$, $\text{Var}(Y) = \phi$

B. $\mathbb{E}[Y] = b''(\theta)$, $\text{Var}(Y) = \phi b'(\theta)$

C. $\mathbb{E}[Y] = b'(\theta) = \theta$, $\text{Var}(Y) = a(\phi)b''(\theta) = \phi$

D. $\mathbb{E}[Y] = \mu$, $\text{Var}(Y) = \mu^2$

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q5. Que valent les moments de Y ?

Réponses correctes : A et C

$\mathbb{E}[Y] = b'(\theta) = \theta = \mu$, $\mathsf{Var}(Y) = a(\phi) \cdot b''(\theta) = \phi$

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q6. La régression linéaire est-elle un GLM ?

A. GLM avec loi normale, lien log

B. GLM avec loi normale, lien identité

C. Ce n'est pas un GLM car Y est continu

D. GLM avec fonction de lien logit

Exercice : La Loi Normale $\in$ Famille Exponentielle ?

Q6. La régression linéaire est-elle un GLM ?
Réponse correcte : B
Régression linéaire = GLM avec loi normale et la fonction identité comme lien.

Solution de l'Exercice : Montrons que la loi normale de $Y \sim \mathcal{N}(\mu, \sigma^2)$ appartient à la famille exponentielle des densités.

Solution : La Loi Normale $\in$ Famille Exponentielle ?

- Montrons que la loi normale de $Y \sim \mathcal{N}(\mu, \sigma^2)$ **appartient à la famille exponentielle des densités:**

$$f_Y(y;\theta) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right)$$

- **Étape 1 : Écrire la densité de la loi normale**

$$f_Y(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

- **Étape 2 : Mise sous forme exponentielle**

$$\begin{aligned} f_Y(y) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{y^2 - 2y\mu + \mu^2}{2\sigma^2}\right) \\ &= \exp\left(-\frac{y^2}{2\sigma^2} + \frac{y\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right) \\ &= \exp\left(\frac{y\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} - \frac{y^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right) \end{aligned}$$

Solution : La Loi Normale $\in$ Famille Exponentielle ?

- **Étape 2 : Mise sous forme exponentielle**

$$
\begin{aligned}
f_Y(y) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{y^2 - 2y\mu + \mu^2}{2\sigma^2}\right) \\
&= \exp\left(-\frac{y^2}{2\sigma^2} + \frac{y\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right) \\
&= \exp\left(\frac{y\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} - \frac{y^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right) \\
&= \exp\left(\frac{y\mu - \mu^2/2}{\sigma^2} - \frac{y^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right)
\end{aligned}
$$

On reconnaît :

$$
f_Y(y) = \exp\left(\underbrace{\frac{y\mu - \mu^2/2}{\sigma^2}}_{\frac{y\theta - b(\theta)}{a(\phi)}} + \underbrace{-\frac{y^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)}_{c(y,\phi)}\right)
$$

Solution : La Loi Normale $\in$ Famille Exponentielle ?

- **Étape 3 : Identification des termes** On reconnaît :

$$f_Y(y) = \exp\left(\underbrace{\frac{y\mu - \mu^2/2}{\sigma^2}}_{\frac{y\theta - b(\theta)}{a(\phi)}} + \underbrace{-\frac{y^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)}_{c(y,\phi)} \right)$$

- **Identification :**
  - $\theta = \mu$: le paramètre canonique est l'espérance μ.
  - $a(\phi) = \phi = \sigma^2$, $b(\theta) = \frac{1}{2}\theta^2$.
  - $c(y, \phi) = -\frac{y^2}{2\phi} - \frac{1}{2}\log(2\pi\phi)$.
  - Fonction de lien canonique : $g(\mu) = \theta = \mu \Rightarrow g(\mu) = \mu$.
- **Donc :**
  - Le lien est l'identité : $\mu = \eta$
  - Le prédicteur linéaire est $\eta = \mathbf{x}_i^\top \boldsymbol{\beta} \Rightarrow \mu = \mathbf{x}_i^\top \boldsymbol{\beta}$

Conclusion : La régression linéaire est un GLM avec :

$$Y_i \sim \mathcal{N}(\mu_i, \sigma^2), \quad \mu_i = \mathbf{x}_i^\top \boldsymbol{\beta}$$

Solution : La Loi Normale $\in$ Famille Exponentielle ?

Composante du GLM	**Régression linéaire**
Distribution de Y	$Y \sim \mathcal{N}(\mu, \sigma^2)$
Paramètre naturel θ	$\theta = \mu$
Paramètre de dispersion ϕ	$\phi = \sigma^2$
Fonction de lien $g(\mu)$	$g(\mu) = \mu$ (identité)
Prédicteur linéaire η	$\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$
Lien entre μ_i et η_i	$\mu_i = \eta_i \Rightarrow \mu = \mathbf{x}_i^\top \boldsymbol{\beta}$
Méthode d'estimation	Maximum de vraisemblance (OLS)

Conclusion : La régression linéaire *est* un GLM avec loi normale, lien identité, estimation par moindres carrés.

Exercice : Montrons que la loi de Poisson $Y \sim \mathcal{P}(\lambda)$ appartient à la famille exponentielle.

Exercice : La Loi de Poisson $\in$ Famille Exponentielle ?

Q1. Quelle est la fonction de masse de $Y \sim \mathcal{P}(\lambda)$?

A. $f_Y(y) = \lambda^y \exp(-\lambda)$

B. $f_Y(y) = \dfrac{\lambda^y}{y!} \exp(-\lambda)$

C. $f_Y(y) = \exp(-\lambda y)$

D. $f_Y(y) = \dfrac{1}{y!} \exp(\lambda^y - \lambda)$

Exercice : La Loi de Poisson $\in$ Famille Exponentielle ?

Q1. Quelle est la fonction de masse de $Y \sim \mathcal{P}(\lambda)$?
Réponse correcte : B
La fonction de masse est :

$$f_Y(y) = \frac{\lambda^y}{y!} \exp(-\lambda), \quad y \in \mathbb{N}$$

Exercice : La Loi de Poisson $\in$ Famille Exponentielle ?

Q2. Mise sous forme exponentielle : quelle est la bonne forme ?

A. $\exp\left(y \log \lambda - \lambda - \log y!\right)$

B. $\exp\left(\lambda y - \lambda^2 - \log y!\right)$

C. $\exp\left(y\lambda - \lambda - y^2\right)$

D. $\exp\left(y + \lambda - y \log y\right)$

Exercice : La Loi de Poisson $\in$ Famille Exponentielle ?

Q2. Mise sous forme exponentielle : quelle est la bonne forme ?
Réponse correcte : A
On reconnaît la forme exponentielle :

$$f_Y(y) = \exp\left(y \log \lambda - \lambda - \log y!\right)$$

Exercice : La Loi de Poisson $\in$ Famille Exponentielle ?

Q3. Quels sont les bons choix pour les composantes du modèle exponentiel ?

A. $\theta = \log \lambda$, $b(\theta) = e^{\theta}$, $a(\phi) = 1$

B. $\theta = \lambda$, $b(\theta) = \log \theta$, $a(\phi) = \phi = \lambda$

C. $\theta = \lambda$, $b(\theta) = \theta$, $a(\phi) = 1$

D. $\theta = \log \lambda$, $b(\theta) = \theta^2$, $a(\phi) = \lambda$

Exercice : La Loi de Poisson $\in$ Famille Exponentielle ?

Q3. Quels sont les bons choix pour les composantes du modèle exponentiel ?

Réponse correcte : A

En effet, on a :

$$\boxed{\theta = \log \lambda, \quad b(\theta) = e^{\theta} = \lambda, \quad a(\phi) = 1}$$

Exercice : La Loi de Poisson $\in$ Famille Exponentielle ?

Q4. Que valent les moments de $Y \sim \mathcal{P}(\lambda)$?

A. $\mathbb{E}[Y] = \theta$, $\text{Var}(Y) = \theta$

B. $\mathbb{E}[Y] = b'(\theta) = e^{\theta}$, $\text{Var}(Y) = b''(\theta) = e^{\theta}$

C. $\mathbb{E}[Y] = b''(\theta)$, $\text{Var}(Y) = a(\phi)b'(\theta)$

D. $\mathbb{E}[Y] = \log(\lambda)$, $\text{Var}(Y) = \lambda^2$

Exercice : La Loi de Poisson $\in$ Famille Exponentielle ?

Q4. Que valent les moments de $Y \sim \mathcal{P}(\lambda)$?
Réponse correcte : B
On a :

$$\mathbb{E}[Y] = b'(\theta) = e^{\theta} = \lambda, \quad \text{Var}(Y) = b''(\theta) = e^{\theta} = \lambda$$

Exercice : La Loi de Poisson $\in$ Famille Exponentielle ?

Q5. La régression de Poisson est-elle un GLM ?

A. GLM avec loi de Poisson, lien identité

B. GLM avec loi de Poisson, lien carré

C. GLM avec loi de Poisson, lien log

D. Ce n'est pas un GLM car Y est discrète

Exercice : La Loi de Poisson $\in$ Famille Exponentielle ?

Q5. La régression de Poisson est-elle un GLM ?
Réponse correcte : C
Lien canonique de la loi de Poisson : $g(\mu) = \log \mu$ (car $\theta = \log \lambda = \log \mu$)

Solution de l'Exercice : Montrons que la loi de Poisson $Y \sim \mathcal{P}(\lambda)$ appartient à la famille exponentielle.

Solution : La Loi de Poisson $\in$ Famille Exponentielle ?

- **Étape 1 : Écrire la densité de la loi de Poisson $Y \sim \mathcal{P}(\lambda)$:**

$$f_Y(y) = \frac{\lambda^y}{y!} e^{-\lambda}, \quad y \in \mathbb{N}$$

- **Étape 2 : Mettre l'expression sous forme exponentielle :**

$$f_Y(y) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right)$$

- On écrit alors:

$$f_Y(y) = \frac{\lambda^y}{y!} e^{-\lambda} = \exp\left(y \log \lambda - \lambda - \log y!\right)$$

- **Étape 3 : Identification des termes**
  Cela ressemble à :

$$f_Y(y) = \exp\left(y\theta - b(\theta) + c(y)\right),$$

  avec :

$$\theta = \log \lambda, \quad b(\theta) = e^\theta = \lambda, \quad c(y) = -\log y!$$

Solution : Moments

- On vient d'exprimer $f_Y(y)$ sous forme exponentielle où l'on a:

$$f_Y(y) = \exp\left(y\theta - b(\theta) + c(y)\right)$$

avec :

$$\theta = \log\lambda, \quad b(\theta) = e^\theta = \lambda, \quad c(y) = -\log y!$$

- Dans la forme canonique exponentielle, on a :

$$\mathbb{E}[Y] = b'(\theta), \quad \text{Var}(Y) = a(\phi)\cdot b''(\theta)$$

- Or :

$$b(\theta) = e^\theta \Rightarrow b'(\theta) = e^\theta = \lambda, \quad b''(\theta) = e^\theta = \lambda$$

- Donc :

$$\mathbb{E}[Y] = \lambda, \quad \text{Var}(Y) = 1\cdot\lambda = \lambda$$

Solution : La Loi de Poisson $\in$ Famille Exponentielle ?

- **Étape 3 : Identification des termes**
  On reconnaît :

$$f_Y(y) = \exp\left(\underbrace{y\log\lambda - \lambda}_{y\theta - b(\theta)} + \underbrace{-\log y!}_{c(y)}\right)$$

- **Identification :**
  - $\theta = \log\lambda$: le paramètre canonique est le logarithme de l'espérance.
  - $b(\theta) = e^\theta = \lambda$
  - $a(\phi) = 1$, car il n'y a pas de paramètre de dispersion dans le modèle de Poisson.
  - $c(y, \phi) = -\log y!$
  - $\mathbb{E}[Y] = \lambda = \mu$.
  - Fonction de lien canonique :
    $g(\mu) = \theta = \log\lambda = \log\mu \Rightarrow g(\mu) = \log\mu$
- **Donc :**
  - Le lien est logarithmique : $\log\mu = \eta$
  - Le prédicteur linéaire est : $\eta = \mathbf{x}_i^\top\boldsymbol{\beta} \Rightarrow \mu = \exp(\mathbf{x}_i^\top\boldsymbol{\beta})$.

Récapitulatif : Poisson = GLM

Composante du GLM	**Régression de Poisson**
Distribution de Y	$Y \sim \mathcal{P}(\mu)$
Paramètre naturel θ	$\theta = \log \mu$
Paramètre de dispersion ϕ	$\phi = 1$
Fonction de lien $g(\mu)$	$g(\mu) = \log \mu$
Prédicteur linéaire η	$\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$
Lien entre μ_i et η_i	$\mu_i = \exp(\eta_i) \Rightarrow \mu_i = \exp(\mathbf{x}_i^\top \boldsymbol{\beta})$
Méthode d'estimation	Maximum de vraisemblance

Conclusion : La régression de Poisson est un GLM avec lien log et distribution de Poisson :

$$Y_i \sim \mathcal{P}(\mu_i), \quad \mu_i = \exp(\mathbf{x}_i^\top \boldsymbol{\beta}).$$

Exercice : Estimation par Maximum de Vraisemblance pour la Régression de Poisson

QCM 1 : MV pour la Régression de Poisson

Question : Quelle est l'expression correcte de la fonction de masse de probabilité pour une variable aléatoire $Y_i \sim \mathcal{P}(\mu_i)$?

A. $\mathbb{P}(Y_i = y_i) = \mu_i^{y_i} \exp(-\mu_i)$

B. $\mathbb{P}(Y_i = y_i) = \frac{\mu_i^{y_i} \exp(-\mu_i)}{y_i!}$

C. $\mathbb{P}(Y_i = y_i) = \frac{y_i^{\mu_i} \exp(-y_i)}{\mu_i!}$

D. $\mathbb{P}(Y_i = y_i) = \mu_i \cdot \exp(-y_i)$

QCM 1 : MV pour la Régression de Poisson

Question : Quelle est l'expression correcte de la fonction de masse de probabilité pour une variable aléatoire $Y_i \sim \mathcal{P}(\mu_i)$?
Bonne réponse : B

Justification : C'est la définition exacte de la loi de Poisson :

$$\mathbb{P}(Y_i = y_i) = \frac{\mu_i^{y_i} \exp(-\mu_i)}{y_i!}$$

QCM 2 : MV pour la Régression de Poisson

Question : Dans le modèle de régression de Poisson, quel lien exprime correctement l'espérance μ_i en fonction des variables explicatives $\mathbf{x}_i$?

A. $\mu_i = \mathbf{x}_i^\top \boldsymbol{\beta}$

B. $\mu_i = \log(\mathbf{x}_i^\top \boldsymbol{\beta})$

C. $\log(\mu_i) = \mathbf{x}_i^\top \boldsymbol{\beta}$

D. $\mu_i = \sqrt{\mathbf{x}_i^\top \boldsymbol{\beta}}$

QCM 2 : MV pour la Régression de Poisson

Question : Dans le modèle de régression de Poisson, quel lien exprime correctement l'espérance μ_i en fonction des variables explicatives $\mathbf{x}_i$?
Bonne réponse : C

Justification : C'est la fonction de lien canonique utilisée pour les modèles de régression de Poisson.

QCM 3 : MV pour la Régression de Poisson

Question : Quelle est la forme correcte de la log-vraisemblance dans le modèle de régression de Poisson (à un facteur constant près) ?

A. $\sum_{i=1}^{n} [y_i \eta_i - \exp(\eta_i)]$
B. $\sum_{i=1}^{n} [y_i \cdot \log(\eta_i) - \eta_i]$
C. $\sum_{i=1}^{n} [y_i \cdot \exp(\eta_i) - \eta_i]$
D. $\sum_{i=1}^{n} [y_i^2 - \eta_i]$

QCM 3 : MV pour la Régression de Poisson

Question : Quelle est la forme correcte de la log-vraisemblance dans le modèle de régression de Poisson (à un facteur constant près) ?
Bonne réponse : A

Justification : La log-vraisemblance est donnée (à un terme constant près) par :

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i \eta_i - \exp(\eta_i)]$$

QCM 4a : MV pour la Régression de Poisson

Question : Trouvez $\frac{\partial}{\partial \boldsymbol{\beta}} \mathbf{x}_i^\top \boldsymbol{\beta}$.

A. $\boldsymbol{\beta}$

B. $\mathbf{x}_i$

C. $\mathbf{x}_i^\top$

D. 1

QCM 4a : MV pour la Régression de Poisson

Question : Trouvez $\frac{\partial}{\partial \boldsymbol{\beta}} \mathbf{x}_i^\top \boldsymbol{\beta}$.
Bonne réponse : B

Justification : La dérivée de $\mathbf{x}_i^\top \boldsymbol{\beta}$ par rapport à $\boldsymbol{\beta}$ est le vecteur $\mathbf{x}_i$, car c'est une fonction linéaire.

QCM 4b : MV pour la Régression de Poisson

Question : Trouvez $\frac{\partial}{\partial \boldsymbol{\beta}} \exp(\mathbf{x}_i^\top \boldsymbol{\beta})$.

A. $\mathbf{x}_i$

B. $\exp(\mathbf{x}_i^\top \boldsymbol{\beta})$

C. $\exp(\mathbf{x}_i^\top \boldsymbol{\beta}) \cdot \mathbf{x}_i$

D. $\mathbf{x}_i^\top \cdot \exp(\boldsymbol{\beta})$

QCM 4b : MV pour la Régression de Poisson

Question : Trouvez $\frac{\partial}{\partial \boldsymbol{\beta}} \exp(\mathbf{x}_i^\top \boldsymbol{\beta})$.
Bonne réponse : C

Justification : C'est une application de la règle de la chaîne :

$$\frac{\partial}{\partial \boldsymbol{\beta}} \exp(\mathbf{x}_i^\top \boldsymbol{\beta}) = \exp(\mathbf{x}_i^\top \boldsymbol{\beta}) \cdot \mathbf{x}_i$$

QCM 4c : MV pour la Régression de Poisson

Question : Quelle est l'expression correcte du gradient de la log-vraisemblance par rapport à $\boldsymbol{\beta}$, $\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$?

A. $\sum_{i=1}^{n} y_i \cdot \mathbf{x}_i$
B. $\sum_{i=1}^{n} \exp(\mathbf{x}_i^\top \boldsymbol{\beta}) \cdot \mathbf{x}_i$
C. $\sum_{i=1}^{n} (y_i - \exp(\mathbf{x}_i^\top \boldsymbol{\beta})) \cdot \mathbf{x}_i$
D. $\sum_{i=1}^{n} (y_i + \exp(\mathbf{x}_i^\top \boldsymbol{\beta})) \cdot \mathbf{x}_i$

QCM 4c : MV pour la Régression de Poisson

Question : Quelle est l'expression correcte du gradient de la log-vraisemblance par rapport à $\boldsymbol{\beta}$, $\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$?
Bonne réponse : C

Justification : En combinant les dérivées des deux termes :

$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} \left[y_i - \exp(\mathbf{x}_i^\top \boldsymbol{\beta}) \right] \mathbf{x}_i$$

QCM 5 : MV pour la Régression de Poisson

Question : Pourquoi ne peut-on pas obtenir de solution fermée pour $\hat{\boldsymbol{\beta}}$ dans le modèle de régression de Poisson ?

A. Car la log-vraisemblance dépend de $\log(y_i!)$

B. Car les dérivées impliquent des fonctions exponentielles

C. Car les variables $\mathbf{x}_i$ sont corrélées

D. Car y_i peut prendre des valeurs nulles

QCM 5 : MV pour la Régression de Poisson

Question : Pourquoi ne peut-on pas obtenir de solution fermée pour $\hat{\boldsymbol{\beta}}$ dans le modèle de régression de Poisson ?
Bonne réponse : B

Justification : Les termes exponentiels dans le gradient empêchent de résoudre explicitement le système $\nabla \ell(\boldsymbol{\beta}) = 0$.

Solution : Estimation par Maximum de Vraisemblance pour la Régression de Poisson

Solution : MV pour la Régression de Poisson

- **Loi de Poisson (définition)** : Soit une variable aléatoire $Y_i \sim \mathcal{P}(\mu_i)$, la fonction de masse de probabilité est :

$$\mathbb{P}(Y_i = y_i \mid \mu_i) = \frac{\mu_i^{y_i} \exp(-\mu_i)}{y_i!}, \quad y_i \in \mathbb{N}$$

- **Lien avec les variables explicatives : Régression de Poisson** Dans un modèle de régression de Poisson (GLM), on relie l'espérance μ_i aux variables explicatives via une fonction de lien canonique :

$$\log(\mu_i) = \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta} \quad \Rightarrow \quad \mu_i = \exp(\mathbf{x}_i^\top \boldsymbol{\beta})$$

- **Substitution dans la densité :** On remplace alors μ_i dans la loi de Poisson par $\exp(\mathbf{x}_i^\top \boldsymbol{\beta})$, ce qui donne :

$$\mathbb{P}(Y_i = y_i \mid \mathbf{x}_i) = \frac{\left[\exp(\mathbf{x}_i^\top \boldsymbol{\beta})\right]^{y_i} \exp\left(-\exp(\mathbf{x}_i^\top \boldsymbol{\beta})\right)}{y_i!}$$

- **Utilité** : Cette expression est la base de la construction de la fonction de vraisemblance pour le modèle.

Solution : MV pour la Régression de Poisson

- On a donc :

$$\mathbb{P}(Y_i = y_i \mid \mathbf{x}_i) = \frac{[\exp(\mathbf{x}_i^\top \boldsymbol{\beta})]^{y_i} \exp(-\exp(\mathbf{x}_i^\top \boldsymbol{\beta}))}{y_i!}$$

- Hypothèse : observations indépendantes $\Rightarrow$ vraisemblance :

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{n} \frac{[\exp(\mathbf{x}_i^\top \boldsymbol{\beta})]^{y_i} \exp(-\exp(\mathbf{x}_i^\top \boldsymbol{\beta}))}{y_i!}$$

- On prend la fonction log-vraisemblance :

$$\ell(\boldsymbol{\beta}) = \log \mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \exp(\mathbf{x}_i^\top \boldsymbol{\beta}) - \log(y_i!) \right]$$

- La fonction objectif à maximiser est alors donnée par :

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i \eta_i - \exp(\eta_i) - \log(y_i!) \right], \quad \text{où } \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$$

- Le terme $\log(y_i!)$ ne dépend pas de $\boldsymbol{\beta}$, on peut l'ignorer pour l'optimisation.

Solution : MV pour la Régression de Poisson

- On fonction log-vraisemblance pour des observations indépendantes est donnée par :

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i \cdot \mathbf{x}_i^\top \boldsymbol{\beta} - \exp(\mathbf{x}_i^\top \boldsymbol{\beta}) - \log(y_i!) \right].$$

- **Calcul du gradient :** dérivons chaque terme par rapport à $\boldsymbol{\beta}$:

$$\frac{\partial}{\partial \boldsymbol{\beta}} \left(y_i \cdot \mathbf{x}_i^\top \boldsymbol{\beta} \right) = y_i \cdot \mathbf{x}_i.$$

$$\frac{\partial}{\partial \boldsymbol{\beta}} \left(\exp(\mathbf{x}_i^\top \boldsymbol{\beta}) \right) = \exp(\mathbf{x}_i^\top \boldsymbol{\beta}) \cdot \mathbf{x}_i.$$

$$\frac{\partial}{\partial \boldsymbol{\beta}} \left(\log(y_i!) \right) = 0 \quad \text{(constante)}.$$

- **Donc, le gradient de la log-vraisemblance est :**

$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} \left[y_i - \exp(\mathbf{x}_i^\top \boldsymbol{\beta}) \right] \mathbf{x}_i.$$

Solution : MV pour la Régression de Poisson

- Contrairement à la régression linéaire (OLS), il n'y a pas de solution analytique pour :

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta}).$$

- Ceci est dû au fait qu'on ne peut pas résoudre :

$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbf{0}.$$

de manière explicite car les dérivées incluent des exponentielles :

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} [y_i - \mu_i] \mathbf{x}_i, \quad \text{avec } \mu_i = \exp(\mathbf{x}_i^\top \boldsymbol{\beta}).$$

- $\Rightarrow$ **Il faut utiliser une méthode numérique d'optimisation.**
- On peut utiliser la **descente de gradient** pour trouver $\hat{\boldsymbol{\beta}}$.

Solution : MV pour la Régression de Poisson

- On maximise $\ell(\boldsymbol{\beta})$ en ajustant $\boldsymbol{\beta}$ dans la direction du **gradient** (vecteur score) :

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \alpha \cdot \nabla \ell(\boldsymbol{\beta}^{(t)}).$$

- Le gradient de la log-vraisemblance est donné par:

$$\nabla \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - \exp(\mathbf{x}_i^\top \boldsymbol{\beta})\right) \mathbf{x}_i = \mathbf{X}^\top (\mathbf{y} - \boldsymbol{\mu}).$$

- Où $\boldsymbol{\mu} = \exp(\mathbf{X}\boldsymbol{\beta})$ est le vecteur des espérances.
- On met à jour $\boldsymbol{\beta}$ jusqu'à convergence.

Solution : MV pour la Régression de Poisson

- **Pourquoi recourir à la descente de gradient?**
- **Pas de solution analytique :** les équations ne se simplifient pas comme pour OLS.
- **Fonction objectif concave :** (la log-vraisemblance est concave pour la régression de Poisson) $\Rightarrow$ toute méthode de gradient converge vers le maximum global.
- **Méthode simple et générale :** fonctionne pour d'autres GLMs (logistique, gamma, etc).
- D'autres alternatives peuvent être efficaces :
  - Méthode de Newton-Raphson
  - Fisher scoring
  - IRLS (Iteratively Reweighted Least Squares)

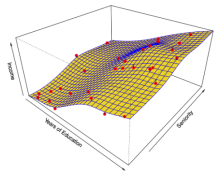
Exercice : Implémentation de la Régression de Poisson en Python sur le Jeu de Donnée *Carseats*

Exercice : Implémentation de la Régression de Poisson

- Dans cet exercice, vous implémenterez manuellement les composantes essentielles d'un modèle de régression de Poisson pour prédire les ventes de sièges pour enfants à partir de variables socio-économiques et commerciales.
- Le jeu de données `Carseats`, issu du manuel *An Introduction to Statistical Learning* (ISLP), contient les caractéristiques de 400 magasins vendant des sièges pour enfants. Chaque ligne correspond à un magasin.
- Le jeu de données vous est fourni sur Moodle. Vous pouvez aussi le télécharger en utilisant la librairie ISLP (`!pip install ISLP`).

Exercice : Implémentation de la Régression de Poisson

- Les variables incluent :
  - `Sales` : Ventes de sièges auto pour enfants (en **milliers d'unités**).
  - `CompPrice`, `Price` : Prix chez le concurrent et prix du magasin.
  - `Income`, `Population` : Revenu moyen, population locale.
  - `Advertising`, `Education`, `Age` : Informations socio-démographiques.
  - `ShelveLoc` : Emplacement du produit (`Good`, `Medium`, `Bad`).
  - `Urban`, `US` : Zone urbaine ? Localisé aux États-Unis ?



Install instructions

Datasets used in ISLP

- Auto Data
- Bike sharing data
- Boston Data
- Brain Cancer Data
- Caravan
- Sales of Child Car Seats
- U.S. News and World Report's College Data
- Credit Card Balance Data
- Credit Card Default Data
- Fund Manager Data
- Baseball Data
- Khan Gene Data
- NCI 60 Data
- New York Stock Exchange Data
- Orange Juice Data

Sales of Child Car Seats

A simulated data set containing sales of child car seats at 400 different stores.

- `Sales` : Unit sales (in thousands) at each location
- `CompPrice` : Price charged by competitor at each location
- `Income` : Community income level (in thousands of dollars)
- `Advertising` : Local advertising budget for company at each location (in thousands of dollars)
- `Population` : Population size in region (in thousands)
- `Price` : Price company charges for car seats at each site
- `ShelveLoc` : A factor with levels Bad, Good and Medium indicating the quality of the shelving location for the car seats at each site
- `Age` : Average age of the local population
- `Education` : Education level at each location
- `Urban` : A factor with levels No and Yes to indicate whether the store is in an urban or rural location
- `US` : A factor with levels No and Yes to indicate whether the store is in the US or not

```
from ISLP import load_data
Carseats = load_data('Carseats')
Carseats.columns
```

```
Index(['Sales', 'CompPrice', 'Income', 'Advertising', 'Population', 'Price',
       'ShelveLoc', 'Age', 'Education', 'Urban', 'US'],
      dtype='object')
```

```
Carseats.shape
```

Exercice : Implémentation de la Régression de Poisson

- Vous devez compléter du code Python dans lequel :
  - Vous implémenterez la log-vraisemblance pour la régression de Poisson.
  - Vous implémenterez le gradient de cette log-vraisemblance.
  - Vous implémenterez la fonction d'entraînement `poisson_regression` qui met à jour les coefficients $\boldsymbol{\beta}$ par descente de gradient.
- La variable cible `Sales` a été arrondie à l'entier le plus proche.
- Les variables catégorielles ont été encodées avec des indicatrices.
- Les données sont divisées en train (80%) et validation (20%).
- La standardisation est faite après le split en utilisant uniquement le train.
- Vous utiliserez la descente de gradient pour maximiser la log-vraisemblance.
- Le clipping est appliqué à $\eta = \mathbf{X}\boldsymbol{\beta}$ pour éviter les overflows numériques.
- Un graphique montrera l'évolution de la log-vraisemblance pour les ensembles d'entraînement et de validation.

Exercice : Implémentation de la Régression de Poisson

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# --- Charger les données Carseats ---
df = pd.read_csv("Carseats_raw.csv")

# --- Transformation de la variable cible ---
df['Sales'] = df['Sales'].round().astype(int)

# --- Encodage des variables catégorielles ---
df_encoded = pd.get_dummies(df, drop_first=False)

# --- Variables explicatives et cible ---
X_vars = [col for col in df_encoded.columns if col != 'Sales']
X_raw = df_encoded[X_vars].values
y = df_encoded['Sales'].values

# --- Division train / validation AVANT standardisation ---
X_raw_train, X_raw_val, y_train, y_val = train_test_split(X_raw, y, test_size=0.2, random_state=8302)

# --- Standardisation basée uniquement sur le train ---
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_raw_train)
X_val_std = scaler.transform(X_raw_val)

# --- Ajout de l'intercept ---
X_train = np.column_stack((np.ones(X_train_std.shape[0]), X_train_std))
X_val = np.column_stack((np.ones(X_val_std.shape[0]), X_val_std))
variables = ['Intercept'] + X_vars
```

Exercice : Implémentation de la Régression de Poisson

```python
# --- Fonctions log-vraisemblance et gradient ---
def log_likelihood(beta, X, y):
    eta = _______
    eta = np.clip(eta, a_max=30, a_min=None)
    mu = _______
    return _______

def gradient(beta, X, y):
    eta = _______
    eta = np.clip(eta, a_max=30, a_min=None)
    mu = _______
    return _______

# --- Descente de gradient ---
def poisson_regression(X, y, lr=1e-5, max_iter=1000, tol=1e-6, X_val=None, y_val=None):
    beta = np.zeros(X.shape[1])
    ll_history_train = []
    ll_history_val = []

    for iteration in range(max_iter):
        grad = gradient(_______)
        beta = _______

        ll_train = log_likelihood(beta, X, y)
        ll_history_train.append(ll_train)

        if X_val is not None and y_val is not None:
            ll_val = log_likelihood(beta, _______, _______)
            ll_history_val.append(ll_val)

        if iteration > 0 and abs(ll_history_train[-1] - ll_history_train[-2]) < tol:
            print(f"Convergence atteinte en {iteration} itérations.")
            break

    return beta, ll_history_train, ll_history_val
```

Exercice : Implémentation de la Régression de Poisson

```python
# --- Entraînement ---
beta_hat, history_train, history_val = poisson_regression(X_train, y_train, X_val=_______, y_val=_______, lr=1e-5)

# --- Affichage des coefficients estimés ---
print("Coefficients estimés :")
for var, coef in zip(variables, beta_hat):
    print(f"  {var} : {coef:.4f}")

# --- Visualisation 1 : Log-vraisemblance normalisée ---
plt.figure(figsize=(8, 5))
plt.plot(np.array(history_train)/len(y_train), label="Train", color="green")
plt.plot(np.array(history_val)/len(y_val), label="Validation", color="orange")
plt.xlabel("Itérations")
plt.ylabel("Log-vraisemblance normalisée")
plt.title("Évolution de la log-vraisemblance")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# --- Visualisation 2 : Prédictions vs Observations ---
mu_pred = np.exp(np.clip(X_val @ beta_hat, a_max=30, a_min=None))

plt.figure(figsize=(8, 5))
plt.scatter(mu_pred, y_val, alpha=0.6, color='royalblue')
plt.plot([min(mu_pred), max(mu_pred)], [min(mu_pred), max(mu_pred)], 'r--')
plt.xlabel("Valeurs prédites (mu)")
plt.ylabel("Valeurs observées (Sales)")
plt.title("Régression de Poisson | Prédictions vs Observations")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Solution: Implémentation de la Régression de Poisson en Python sur le Jeu de Donnée *Carseats*

Solution : Implémentation de la Régression de Poisson

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# --- Charger les données Carseats ---
df = pd.read_csv("Carseats_raw.csv")

# --- Transformation de la variable cible ---
df['Sales'] = df['Sales'].round().astype(int)

# --- Encodage des variables catégorielles ---
df_encoded = pd.get_dummies(df, drop_first=False)

# --- Variables explicatives et cible ---
X_vars = [col for col in df_encoded.columns if col != 'Sales']
X_raw = df_encoded[X_vars].values
y = df_encoded['Sales'].values

# --- Division train / validation AVANT standardisation ---
X_raw_train, X_raw_val, y_train, y_val = train_test_split(X_raw, y, test_size=0.2, random_state=8302)

# --- Standardisation basée uniquement sur le train ---
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_raw_train)
X_val_std = scaler.transform(X_raw_val)

# --- Ajout de l'intercept ---
X_train = np.column_stack((np.ones(X_train_std.shape[0]), X_train_std))
X_val = np.column_stack((np.ones(X_val_std.shape[0]), X_val_std))
variables = ['Intercept'] + X_vars
```

Solution : Implémentation de la Régression de Poisson

```python
# --- Fonctions log-vraisemblance et gradient ---
def log_likelihood(beta, X, y):
    eta = X @ beta
    eta = np.clip(eta, a_max=30, a_min=None)
    mu = np.exp(eta)
    return np.sum(y * eta - mu)

def gradient(beta, X, y):
    eta = X @ beta
    eta = np.clip(eta, a_max=30, a_min=None)
    mu = np.exp(eta)
    return X.T @ (y - mu)

# --- Descente de gradient ---
def poisson_regression(X, y, lr=1e-5, max_iter=1000, tol=1e-6, X_val=None, y_val=None):
    beta = np.zeros(X.shape[1])
    ll_history_train = []
    ll_history_val = []

    for iteration in range(max_iter):
        grad = gradient(beta, X, y)
        beta = beta + lr * grad

        ll_train = log_likelihood(beta, X, y)
        ll_history_train.append(ll_train)

        if X_val is not None and y_val is not None:
            ll_val = log_likelihood(beta, X_val, y_val)
            ll_history_val.append(ll_val)

        if iteration > 0 and abs(ll_history_train[-1] - ll_history_train[-2]) < tol:
            print(f"Convergence atteinte en {iteration} itérations.")
            break

    return beta, ll_history_train, ll_history_val
```

Solution : Implémentation de la Régression de Poisson

```
# --- Entraînement ---
beta_hat, history_train, history_val = poisson_regression(X_train, y_train, X_val=X_val, y_val=y_val, lr=1e-5)

# --- Affichage des coefficients estimés ---
print("Coefficients estimés :")
for var, coef in zip(variables, beta_hat):
    print(f"  {var} : {coef:.4f}")

# --- Visualisation 1 : Log-vraisemblance normalisée ---
plt.figure(figsize=(8, 5))
plt.plot(np.array(history_train)/len(y_train), label="Train", color="green")
plt.plot(np.array(history_val)/len(y_val), label="Validation", color="orange")
plt.xlabel("Itérations")
plt.ylabel("Log-vraisemblance normalisée")
plt.title("Évolution de la log-vraisemblance")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

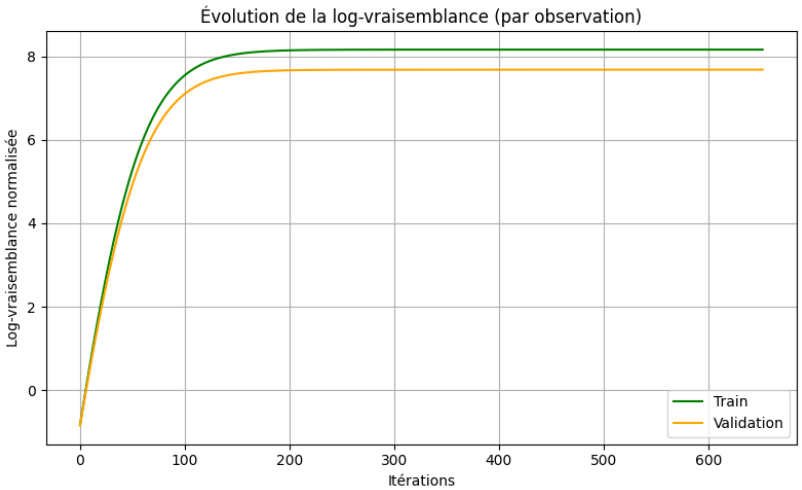
Solution : Implémentation de la Régression de Poisson

```
# --- Visualisation 2 : Prédictions vs Observations ---
mu_pred = np.exp(np.clip(X_val @ beta_hat, a_max=30, a_min=None))
plt.figure(figsize=(8, 5))
plt.scatter(mu_pred, y_val, alpha=0.6, color='royalblue')
plt.plot([min(mu_pred), max(mu_pred)], [min(mu_pred), max(mu_pred)], 'r--')
plt.xlabel("Valeurs prédites (mu)")
plt.ylabel("Valeurs observées (Sales)")
plt.title("Régression de Poisson | Prédictions vs Observations")
plt.grid(True)
plt.tight_layout()
plt.show()
```

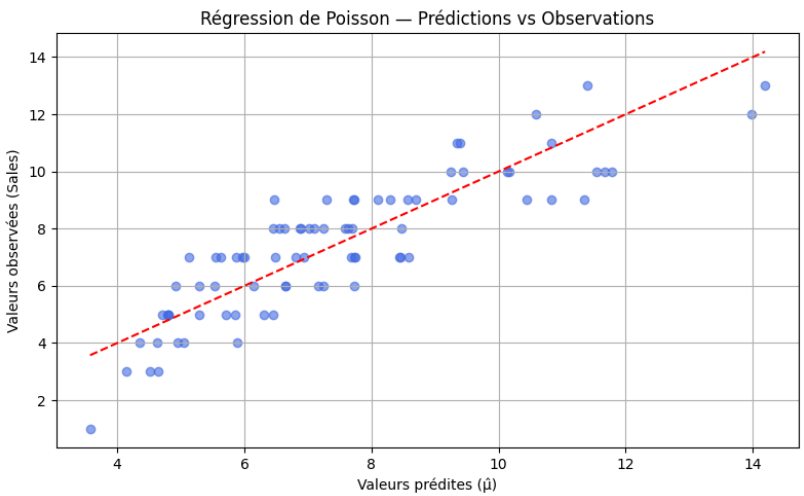
```
Résultat Affiché :
Convergence atteinte en 652 itérations.
Coefficients estimés :
  Intercept : 1.9555
  CompPrice : 0.1939
  Income : 0.0604
  Advertising : 0.0989
  Population : 0.0143
  Price : -0.2953
  Age : -0.1102
  Education : -0.0060
  ShelveLoc_Bad : -0.1294
  ShelveLoc_Good : 0.1335
  ShelveLoc_Medium : 0.0021
  Urban_No : -0.0053
  Urban_Yes : 0.0053
  US_No : 0.0034
  US_Yes : -0.0034
```

Solution : Implémentation de la Régression de Poisson

- La descente de gradient a convergé en 652 itérations, indiquant un bon choix du taux d'apprentissage.
- Les courbes de log-vraisemblance montrent une convergence rapide et l'absence de surapprentissage.
- Les prédictions sur l'ensemble de validation suivent globalement les observations, indiquant une bonne capacité de généralisation.



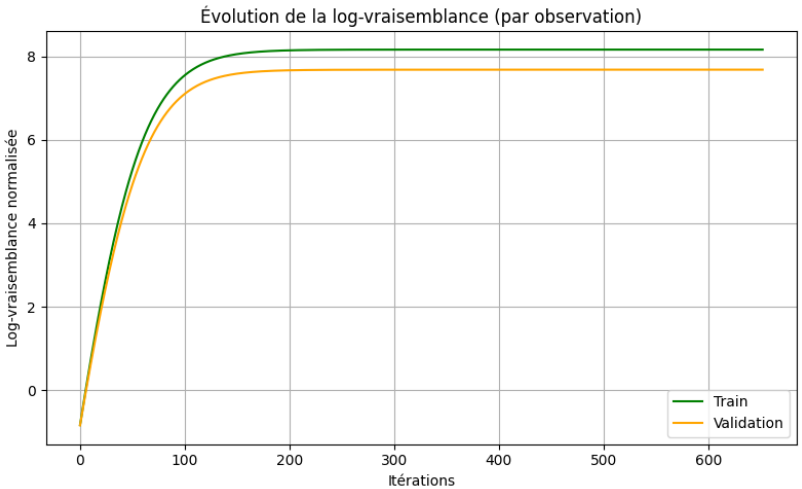
Évolution de la log-vraisemblance



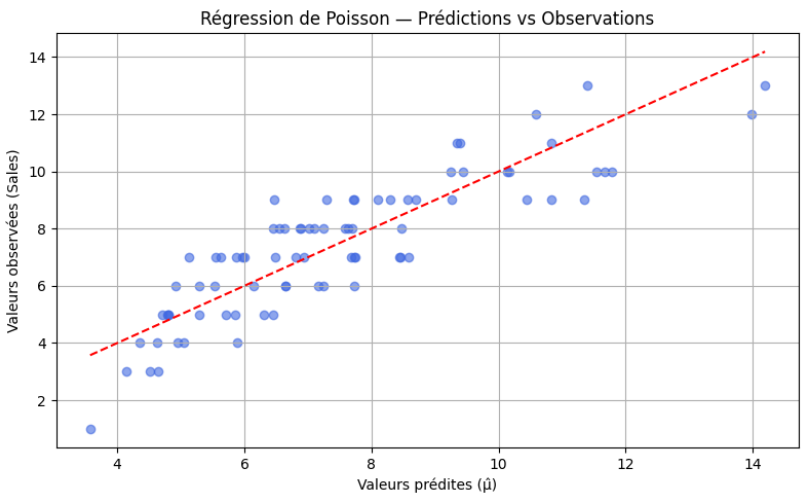
Prédictions vs Observations

Solution : Implémentation de la Régression de Poisson

- Les variables `Price`, `Advertising`, et `ShelveLoc` ont un effet marqué sur les ventes.
- Le coefficient négatif de `Price` est attendu : un prix plus élevé entraîne une baisse des ventes.
- La variable `ShelveLoc_Good` a un effet positif important, illustrant l'impact du placement en rayon.



Évolution de la log-vraisemblance



Prédictions vs Observations

Table des Matières

1 Modèles Linéaires Généralisés

2 Regression Logistique

3 Analyse Linéaire Discriminante

4 Annexe

Regression Logistique

Régression Logistique

Une autre instance des Modèles Linéaires Généralisés (GLMs)

Pourquoi la Régression Logistique ?

- Jusqu'ici, nous avons considéré des variables de sortie $Y \in \mathbb{N}$ (Poisson) ou $Y \in \mathbb{R}$ (Normale).
- **Et si la variable cible est binaire ?**
  Exemple : prédire si un individu est malade ou non, si une transaction est frauduleuse ou non.

$$Y \in \{0, 1\}$$

- **Problème :** La régression linéaire n'est pas adaptée.
  Elle peut produire des prédictions $\hat{y} \notin [0, 1]$, ce qui est incohérent pour une probabilité.
- **Solution :** Utiliser une fonction de lien adaptée :

$$\mu_i = \mathbb{E}[Y_i] = \text{probabilité de succès} \Rightarrow \mu_i \in [0, 1]$$

- C'est ici qu'intervient la **Régression Logistique**, avec fonction de lien logit :

$$g(\mu_i) = \log\left(\frac{\mu_i}{1 - \mu_i}\right) = \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$$

Exercice : Montrons que la loi de Bernoulli $Y \sim \mathcal{B}(p)$ appartient à la famille exponentielle.

Exercice : La Loi de Bernoulli $\in$ Famille Exponentielle ?

Q1. Quelle est la fonction de masse de $Y \sim \mathcal{B}(p)$?

A. $f_Y(y) = \exp(p^y(1-p)^{1-y})$

B. $f_Y(y) = p^y(1-p)^{1-y}$

C. $f_Y(y) = \frac{1}{p^y(1-p)^{1-y}}$

D. $f_Y(y) = y(1-p) + (1-y)p$

Exercice : La Loi de Bernoulli $\in$ Famille Exponentielle ?

Q1. Quelle est la fonction de masse de $Y \sim \mathcal{B}(p)$?
Réponse correcte : B
La fonction de masse de probabilité d'une variable de Bernoulli est bien :

$$f_Y(y) = p^y(1-p)^{1-y}, \quad y \in \{0, 1\}$$

Exercice : La Loi de Bernoulli $\in$ Famille Exponentielle ?

Q2. Mise sous forme exponentielle : lesquelles sont correctes ?

A. $\exp\left(y\log(p) + (1-y)\log(1-p)\right)$

B. $\exp\left(y\log\left(\frac{p}{1-p}\right) + \log(1-p)\right)$

C. $\exp\left(y\log(p) - y\log(1-p)\right)$

D. $\exp\left(y - p\right)$

Exercice : La Loi de Bernoulli $\in$ Famille Exponentielle ?

Q2. Mise sous forme exponentielle : lesquelles sont correctes ?
Réponses correctes : A et B

$$f_Y(y) = \exp\left(y \log\left(\frac{p}{1-p}\right) + \log(1-p)\right)$$

On reconnaît bien la forme exponentielle.

Exercice : La Loi de Bernoulli $\in$ Famille Exponentielle ?

Q3. Identifier les composantes de la forme exponentielle

A. $\theta = \log\left(\frac{p}{1-p}\right), \quad b(\theta) = \log(1+e^{\theta}), \quad a(\phi) = 1$

B. $\theta = p, \quad b(\theta) = p, \quad a(\phi) = 1$

C. $\theta = \frac{1}{p}, \quad b(\theta) = \theta^2, \quad a(\phi) = \theta$

D. $\theta = \log(p), \quad b(\theta) = \log(1+e^{p}), \quad a(\phi) = 1$

Exercice : La Loi de Bernoulli $\in$ Famille Exponentielle ?

Q3. Identifier les composantes de la forme exponentielle
Réponse correcte : A

- Paramètre canonique : $\theta = \log\left(\frac{p}{1-p}\right)$
- Fonction log-partition : $b(\theta) = \log(1 + e^{\theta})$
- Dispersion : $a(\phi) = 1$ (car Bernoulli à variance fixée)

Exercice : La Loi de Bernoulli $\in$ Famille Exponentielle ?

Q4. Quels sont les moments de $Y \sim \mathcal{B}(p)$?

A. $\mathbb{E}[Y] = b'(\theta), \quad \text{Var}(Y) = b''(\theta)$

B. $\mathbb{E}[Y] = p, \quad \text{Var}(Y) = p(1-p)$

C. $\mathbb{E}[Y] = \frac{e^\theta}{1+e^\theta}, \quad \text{Var}(Y) = \frac{e^\theta}{(1+e^\theta)^2}$

D. Toutes les réponses ci-dessus

Exercice : La Loi de Bernoulli $\in$ Famille Exponentielle ?

Q4. Quels sont les moments de $Y \sim \mathcal{B}(p)$?
Réponse correcte : D
Toutes les réponses sont cohérentes selon l'expression $b(\theta) = \log(1 + e^{\theta})$.

$$b'(\theta) = \frac{e^{\theta}}{1 + e^{\theta}}, \quad b''(\theta) = \frac{e^{\theta}}{(1 + e^{\theta})^2}$$

Solution de l'Exercice : Montrons que la loi de Bernoulli $Y \sim \mathcal{B}(\mu)$ appartient à la famille exponentielle.

Rappel : Étapes pour Déterminer $g(\mu)$ et η

- **Étapes à suivre :**
  1. **Écrire la densité dans sa forme classique.**
     Exemple : loi normale, Bernoulli, Poisson...
  2. **Mettre la densité sous forme exponentielle.**
     Identifier les termes permettant d'aboutir à la forme canonique.
  3. **Identifier les composantes du modèle exponentiel :**
     - θ : paramètre naturel
     - ϕ : paramètre de dispersion
     - $a(\phi)$: fonction de pondération
     - $b(\theta)$: fonction log-partition
     - $c(y, \phi)$: fonction de normalisation
  4. **Calculer les moments en fonction de θ :**

$$\mathbb{E}[Y] = b'(\theta), \quad \text{Var}(Y) = a(\phi) \cdot b''(\theta).$$

  5. **Déterminer la fonction de lien canonique $g(\mu)$ en fonction de θ:**

$$\mu = b'(\theta) \Rightarrow g(\mu) = \theta.$$

  6. **Exprimer le prédicteur linéaire du GLM en remplaçant θ par η_i:**

$$\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}, \quad \text{avec } g(\mu_i) = \eta_i$$

Solution : La Loi de Bernoulli $\in$ Famille Exponentielle ?

- Montrons que la loi de Bernoulli $Y \sim \mathcal{B}(\mu)$ **appartient à la famille exponentielle :**

$$f_Y(y;\theta) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right)$$

- **Étape 1 : Écrire la densité de la loi de Bernoulli**

$$f_Y(y) = p^y(1-p)^{1-y}, \quad y \in \{0,1\},\ p \in (0,1)$$

- **Étape 2 : Mise sous forme exponentielle**

$$\begin{aligned} f_Y(y) &= \exp\left(\log(p^y(1-p)^{1-y})\right) \\ &= \exp\left(y\log(p) + (1-y)\log(1-p)\right) \\ &= \exp\left(y\log(p) + \log(1-p) - y\log(1-p)\right) \\ &= \exp\left(y\log\left(\frac{p}{1-p}\right) + \log(1-p)\right) \end{aligned}$$

Solution : La Loi de Bernoulli $\in$ Famille Exponentielle ?

- **Étape 3 : Identification des termes**
  On reconnaît :

$$f_Y(y) = \exp\left(\underbrace{y \log\left(\frac{p}{1-p}\right)}_{y\theta} + \underbrace{\log(1-p)}_{-b(\theta)}\right) = \exp\left(y\theta - b(\theta) + c(y)\right)$$

- **Identification :**
  - $\theta = \log\left(\frac{p}{1-p}\right)$ (paramètre naturel) $\Rightarrow e^\theta = \frac{p}{1-p} \Rightarrow p = \frac{e^\theta}{1+e^\theta}$
  - $b(\theta) = -\log(1-p) = -\log(1 - \frac{e^\theta}{1+e^\theta}) = -\log(\frac{1}{1+e^\theta}) = \log(1+e^\theta)$
  - $a(\phi) = 1$ (pas de paramètre de dispersion)
  - $c(y) = 0$
- **Moments :**

$$\mu = \mathbb{E}[Y] = b'(\theta) = \frac{e^\theta}{1+e^\theta} = p,$$

$$\text{Var}(Y) = a(\phi) \cdot b''(\theta) = \frac{e^\theta}{(1+e^\theta)^2} = p(1-p).$$

Solution : La Loi de Bernoulli $\in$ Famille Exponentielle ?

- On sait que :

$$\mu = \mathbb{E}[Y] = b'(\theta) = \frac{e^\theta}{1+e^\theta} = p$$

- On inverse cette relation pour obtenir θ en fonction de μ :

$$= \frac{e^\theta}{1+e^\theta} \Rightarrow \mu(1+e^\theta) = e^\theta$$
$$\Rightarrow \mu = e^\theta(1-\mu)$$
$$\Rightarrow e^\theta = \frac{\mu}{1-\mu} \Rightarrow \theta = \log\left(\frac{\mu}{1-\mu}\right)$$

- **Conclusion :**

$$\boxed{g(\mu) = \log\left(\frac{\mu}{1-\mu}\right)}$$ (fonction de lien canonique : logit)

- Donc le prédicteur linéaire s'écrit :

$$\eta = \mathbf{x}_i^\top \boldsymbol{\beta}, \quad \mu = p_i = \frac{e^\eta}{1+e^\eta} = \frac{1}{1+e^{-\eta}} = \text{Sigmoïde}(\eta).$$

Solution : La Loi de Bernoulli $\in$ Famille Exponentielle ?

Composante du GLM	**Régression Logistique**
Distribution de Y	$Y \sim \mathcal{B}(\mu)$
Paramètre naturel θ	$\theta = \log\left(\frac{\mu}{1-\mu}\right)$
Paramètre de dispersion ϕ	$\phi = 1$
Fonction de lien $g(\mu)$	$g(\mu) = \log\left(\frac{\mu}{1-\mu}\right)$
Prédicteur linéaire η	$\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$
Lien entre μ_i et η_i	$\mu_i = \frac{1}{1+e^{-\eta_i}} = \text{Sigmoïde}(\eta_i)$
Méthode d'estimation	Maximum de vraisemblance

Conclusion : La régression logistique est un GLM avec une distribution de Bernoulli pour la variable cible et une fonction de lien logit :

$$Y \sim \mathcal{B}(\mu), \quad g(\mu) = \log\left(\frac{\mu}{1-\mu}\right).$$

Exercice : Estimation par Maximum de Vraisemblance pour la Régression Logistique

QCM 1 — Mise sous forme factorisée

Q1. En utilisant $p_i = \frac{1}{1+e^{-\eta_i}}$, laquelle des expressions suivantes correspond à la probabilité :

$$\mathbb{P}(Y_i = y_i \mid \mathbf{x}_i) = p_i^{y_i}(1-p_i)^{1-y_i} \quad \text{mise sous forme factorisée?}$$

A. $\dfrac{e^{y_i \eta_i}}{1+e^{\eta_i}}$

B. $\dfrac{1}{1+e^{-\eta_i}}$

C. $\dfrac{e^{-\eta_i(1+y_i)}}{(1+e^{-\eta_i})}$

D. $\dfrac{1}{e^{-\eta_i}}$

QCM 1 — Mise sous forme factorisée

Bonne réponse : A

Justification :

$$\begin{aligned}\mathbb{P}(Y_i = y_i \mid \mathbf{x}_i) &= \left(\frac{1}{1+e^{-\eta_i}}\right)^{y_i} \left(\frac{e^{-\eta_i}}{1+e^{-\eta_i}}\right)^{1-y_i} \\ &= \frac{e^{-\eta_i(1-y_i)}}{1+e^{-\eta_i}} \\ &= \frac{e^{y_i\eta_i}}{1+e^{\eta_i}}\end{aligned}$$

QCM 2 — Mise sous forme exponentielle finale

Q2. En partant ddu résultat précédent obtenu pour $\mathbb{P}(Y_i = y_i \mid \mathbf{x}_i)$, laquelle des expressions suivantes correspond à sa mise sous forme exponentielle canonique ?

A. $\exp\left(y_i \cdot \eta_i - \log(1 + e^{\eta_i})\right)$

B. $\exp\left(-y_i \cdot \eta_i + \log(1 + e^{\eta_i})\right)$

C. $\exp\left(y_i \cdot \eta_i + \log(1 + e^{-\eta_i})\right)$

D. $\exp\left(-\eta_i + y_i \cdot \eta_i\right)$

QCM 2 — Mise sous forme exponentielle finale

Bonne réponse : A

Justification :

$$\frac{e^{y_i \eta_i}}{1 + e^{\eta_i}} = \exp\left(y_i \cdot \eta_i - \log(1 + e^{\eta_i})\right)$$

QCM 3 — Fonction log-vraisemblance

Q2. En supposant n observations indépendantes, quelle sont les bonnes expressions de la log-vraisemblance $\ell(\boldsymbol{\beta})$ en fonction de η_i ?

A. $\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log\left(\frac{e^{y_i \eta_i}}{1 + e^{\eta_i}}\right)$

B. $\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i \cdot \eta_i - \log(1 + e^{\eta_i})]$

C. $\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i \cdot p_i + (1 - y_i) \log(1 - p_i)]$

D. $\ell(\boldsymbol{\beta}) = \prod_{i=1}^{n} \frac{e^{y_i \eta_i}}{1 + e^{\eta_i}}$

QCM 3 — Fonction log-vraisemblance

Bonne réponse : A et B

Justification :

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log\left(\frac{e^{y_i \eta_i}}{1 + e^{\eta_i}}\right)$$

$$= \sum_{i=1}^{n} [y_i \eta_i - \log(1 + e^{\eta_i})]$$

QCM 4 — Gradient de la log-vraisemblance

Q4. En dérivant la fonction log-vraisemblance, quelle est l'expression correcte du gradient $\nabla \ell(\boldsymbol{\beta})$?

A. $\nabla \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_i \cdot \mathbf{x}_i$

B. $\nabla \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \eta_i) \cdot \mathbf{x}_i$

C. $\nabla \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - p_i) \cdot \mathbf{x}_i$

D. $\nabla \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} p_i \cdot \mathbf{x}_i$

QCM 4 — Gradient de la log-vraisemblance

Bonne réponse : C

Justification :

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i \cdot \eta_i - \log(1 + e^{\eta_i})]$$

$$\Rightarrow \nabla \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - p_i) \cdot \mathbf{x}_i$$

$$\text{avec } p_i = \frac{1}{1 + e^{-\eta_i}} = \text{Sigmoïde}(\eta_i)$$

Solution : Estimation par Maximum de Vraisemblance pour la Régression Logistique

Solution : MV pour la Régression Logistique

- **Loi de Bernoulli (définition)** : Soit $Y_i \sim \mathcal{B}(p_i)$, avec $p_i = \mathbb{P}(Y_i = 1 \mid \mathbf{x}_i) = p_i$. La fonction de densité de probabilité est :

$$\mathbb{P}(Y_i = y_i \mid \mathbf{x}_i) = p_i^{y_i}(1-p_i)^{1-y_i}, \quad y_i \in \{0,1\}$$

- **Lien avec les variables explicatives : Régression logistique** L'espérance $\mu_i = \mathbb{E}[Y_i \mid \mathbf{X}_i = \mathbf{x_i}] = p_i$ est reliée aux variables explicatives via une fonction de lien logit :

$$g(\mu_i) = \log\left(\frac{\mu_i}{1-\mu_i}\right) = \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}.$$

$$\begin{aligned} \Rightarrow \mu_i = \mathbb{E}[Y_i \mid \mathbf{X}_i = \mathbf{x_i}] = \mathbb{P}(Y_i = y_i \mid \mathbf{x}_i) &= \frac{1}{1+e^{-\mathbf{x}_i^\top \boldsymbol{\beta}}} \\ &= \text{Sigmoïde}(\mathbf{x}_i^\top \boldsymbol{\beta}) \\ &= \text{Sigmoïde}(\eta_i). \end{aligned}$$

- On remplace p_i dans la densité :

$$\mathbb{P}(Y_i = y_i \mid \mathbf{x}_i) = \left(\frac{1}{1+e^{-\mathbf{x}_i^\top \boldsymbol{\beta}}}\right)^{y_i} \left(1 - \frac{1}{1+e^{-\mathbf{x}_i^\top \boldsymbol{\beta}}}\right)^{1-y_i}$$

Solution : MV pour la Régression Logistique

- En remplaçant $\mathbf{x}_i^\top \boldsymbol{\beta}$ par η on obtient :

$$\begin{aligned}
\mathbb{P}(Y_i = y_i \mid \mathbf{x}_i) &= \left(\frac{1}{1+e^{-\eta_i}}\right)^{y_i} \left(\frac{e^{-\eta_i}}{1+e^{-\eta_i}}\right)^{1-y_i} \\
&= \frac{e^{-\eta_i(1-y_i)}}{(1+e^{-\eta_i})^{1-y_i+y_i}} \\
&= \frac{e^{-\eta_i+y_i\eta_i}}{1+e^{-\eta_i}} \\
&= \frac{\cancel{e^{-\eta_i}}(e^{y_i\eta_i})}{\cancel{e^{-\eta_i}}(e^{\eta_i}+1)} = \frac{e^{y_i\eta_i}}{1+e^{\eta_i}}, \quad \text{où } \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}.
\end{aligned}$$

- Vraisemblance pour n observations indépendantes :

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{n} \frac{e^{y_i\eta_i}}{1+e^{\eta_i}}, \quad \text{où } \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}.$$

Solution : MV pour la Régression Logistique

- Vraisemblance pour n observations indépendantes :

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i=1}^{n} \frac{e^{y_i \eta_i}}{1 + e^{\eta_i}}, \quad \text{où } \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}.$$

- Fonction log-vraisemblance :

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i \cdot \eta_i - \log(1 + e^{\eta_i}) \right], \quad \text{où } \eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}.$$

- C'est la fonction qu'on va maximiser en fonction de $\boldsymbol{\beta}$:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[y_i \cdot \mathbf{x}_i^\top \boldsymbol{\beta} - \log(1 + e^{\mathbf{x}_i^\top \boldsymbol{\beta}}) \right].$$

Solution : MV pour la Régression Logistique

- **Gradient de la log-vraisemblance :**
  - On dérive chaque terme :

$$\frac{\partial}{\partial \boldsymbol{\beta}} \left(y_i \cdot \mathbf{x}_i^\top \boldsymbol{\beta} \right) = y_i \cdot \mathbf{x}_i$$

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\beta}} \left(\log(1 + e^{\mathbf{x}_i^\top \boldsymbol{\beta}}) \right) &= \frac{e^{\eta_i}}{1 + e^{\eta_i}} \cdot \mathbf{x}_i \\ &= \text{Sigmoïde}(\eta_i) \cdot \mathbf{x}_i \\ &= p_i \cdot \mathbf{x}_i \end{aligned}$$

  - Le gradient total est donc :

$$\nabla \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - p_i)\mathbf{x}_i = \mathbf{X}^\top (\mathbf{y} - \boldsymbol{p}), \quad \text{où} \quad \begin{aligned} p_i &= \frac{1}{1 + e^{-\eta_i}} \\ &= \text{Sigmoïde}(\eta_i). \end{aligned}$$

Solution : MV pour la Régression Logistique

- **Pas de solution analytique :**

$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \mathbf{0} \quad \text{n'a pas de solution explicite.}$$

- Les équations sont non linéaires car :

$$p_i = \frac{1}{1 + e^{-\mathbf{x}_i^\top \boldsymbol{\beta}}} \Rightarrow \text{l'équation est exprimée en fonction de Sigmoïde .}$$

- **Il faut une méthode numérique d'optimisation** : descente de gradient, Newton-Raphson, IRLS...

Solution : MV pour la Régression Logistique

- On met à jour les paramètres avec la descente de gradient :

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \alpha \cdot \nabla \ell(\boldsymbol{\beta}^{(t)})$$

- Rappel : le gradient est :

$$\nabla \ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - p_i)\mathbf{x}_i = \mathbf{X}^\top (\mathbf{y} - \boldsymbol{p})$$

- Où $\boldsymbol{\mu} = \boldsymbol{p} = \left[\frac{1}{1+e^{-\mathbf{x}_1^\top \boldsymbol{\beta}}}, \ldots, \frac{1}{1+e^{-\mathbf{x}_n^\top \boldsymbol{\beta}}}\right]^\top = \text{Sigmoïde}(\mathbf{X}\boldsymbol{\beta})$ est le vecteur des espérances.
- On met à jour $\boldsymbol{\beta}$ jusqu'à convergence.

Pourquoi utiliser la descente de gradient ?

- **Pas de solution analytique** : les dérivées incluent des exponentielles (fonction Sigmoïde).
- **La fonction est concave** : la log-vraisemblance est concave en β, donc toute méthode de gradient converge vers le maximum global.
- **Méthode simple et efficace** : fonctionne pour tous les GLMs.
- Alternatives plus rapides :
  - Méthode de Newton-Raphson
  - Méthode d'IRLS (Iteratively Reweighted Least Squares)
  - Scoring de Fisher

Métriques d'évaluation pour la Régression Logistique

- Une fois le modèle entraîné, il est crucial d'évaluer ses performances.
- Pour un problème de classification binaire (classes 0 et 1), plusieurs métriques sont utilisées.
- On suppose un seuil de décision à 0.5 sauf indication contraire.

Exercice : Implémentation de la Régression Logistique en Python sur le Jeu de Données *Breast Cancer*

Exercice : Implémentation de la Régression Logistique

- Dans cet exercice, vous implémenterez manuellement les composantes essentielles d'un modèle de régression logistique pour prédire si une tumeur mammaire est maligne ou bénigne à partir de mesures cellulaires.
- Le jeu de données `Breast Cancer Wisconsin` est accessible via `sklearn.datasets.load_breast_cancer()`.
- Il contient 569 observations (patients) et 30 variables explicatives continues extraites de l'image d'une cellule.

Exercice : Implémentation de la Régression Logistique

- Les variables incluent notamment :
  - `mean radius`, `mean texture`, `mean perimeter`, `mean area` : mesures moyennes de forme.
  - `mean smoothness`, `mean compactness`, `mean concavity` : régularité des contours.
  - `radius error`, `texture error`, `area error` : erreur-type des mesures.
  - `worst area`, `worst symmetry`, etc. : pires valeurs mesurées sur la tumeur.
- La variable cible est binaire :
  - $y = 1$: Tumeur maligne
  - $y = 0$: Tumeur bénigne

Exercice : Implémentation de la Régression Logistique

- Vous devez compléter un code Python dans lequel :
  - Vous implémenterez la log-vraisemblance pour la régression logistique.
  - Vous implémenterez le gradient de cette log-vraisemblance.
  - Vous implémenterez la fonction `logistic_regression` qui met à jour les coefficients $\boldsymbol{\beta}$ par descente de gradient.
- Les données sont divisées en train (80%) et validation (20%).
- La standardisation est faite après le split, à partir du train seulement.
- Vous utiliserez la descente de gradient pour maximiser la log-vraisemblance.
- Le clipping est appliqué à $\eta = \mathbf{X}\boldsymbol{\beta}$ pour éviter les overflows numériques.
- Un graphique montrera l'évolution de la log-vraisemblance pour les ensembles d'entraînement et de validation.

Exercice : Implémentation de la Régression Logistique

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import seaborn as sns
sns.set_theme()

# --- Charger les données ---
data = load_breast_cancer()
X_raw = data.data
y = data.target
feature_names = data.feature_names

# --- Division train / validation AVANT standardisation ---
X_raw_train, X_raw_val, y_train, y_val = train_test_split(
    X_raw, y, test_size=0.2, random_state=8302
)

# --- Standardisation ---
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_raw_train)
X_val_std = scaler.transform(X_raw_val)

# --- Ajout de l'intercept ---
X_train = np.column_stack((np.ones(X_train_std.shape[0]), X_train_std))
X_val = np.column_stack((np.ones(X_val_std.shape[0]), X_val_std))
variables = ['Intercept'] + list(feature_names)
```

Exercice : Implémentation de la Régression Logistique

```python
# --- Fonctions log-vraisemblance et gradient ---

def sigmoid(z):
    _______

def log_likelihood_logistic(beta, X, y):
    eta = _______
    eta = np.clip(eta, -30, 30)
    p = _______
    return _______

def gradient_logistic(beta, X, y):
    eta = _______
    eta = np.clip(eta, -30, 30)
    p = _______
    return _______

# --- Descente de gradient ---
def logistic_regression(X, y, lr=1e-4, max_iter=10000, tol=1e-6, X_val=None, y_val=None):
    beta = _______ # initialisation de beta
    ll_history_train = _______ # initialisation de la liste des ll pour train
    ll_history_val = _______ # initialisation de la liste des ll pour val
    for iteration in range(max_iter):
        grad = _______
        beta = _______
        ll_train = _______
        ll_history_train.append(_______)
        if X_val is not None and y_val is not None:
            ll_val = log_likelihood_logistic(beta, _______, _______)
            ll_history_val.append(_______)
        if iteration > 0 and abs(ll_history_train[-1] - ll_history_train[-2]) < tol:
            print(f"Convergence atteinte en {iteration} itérations.")
            break
    return beta, ll_history_train, ll_history_val
```

Exercice : Implémentation de la Régression Logistique

```python
# --- Entraînement ---
beta_hat, history_train, history_val = logistic_regression(
    _______, _______, X_val=_______, y_val=_______
)

# --- Affichage des coefficients estimés ---
print("Coefficients estimés :")
for var, coef in zip(variables, beta_hat):
    print(f"  {var} : {coef:.4f}")

# --- Visualisation : Log-vraisemblance ---
plt.figure(figsize=(8, 5))
plt.plot(np.array(history_train)/len(y_train), label="Train", color="green")
plt.plot(np.array(history_val)/len(y_val), label="Validation", color="orange")
plt.xlabel("Itérations")
plt.ylabel("Log-vraisemblance normalisée")
plt.title("Évolution de la log-vraisemblance")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# --- Visualisation : Prédictions vs Observations ---
p_pred = _______

y_pred_class = (p_pred >= _______).astype(int) # Seuil de décision est p = 0.5.
plt.figure(figsize=(8, 5))
plt.scatter(p_pred, y_val, alpha=0.6, color='royalblue')
plt.axvline(x=0.5, color='red', linestyle='--', linewidth=1.5, label='Seuil de décision (0.5)')
plt.xlabel("Probabilité prédite")
plt.ylabel("Classe observée")
plt.title("Régression Logistique | Prédictions vs Observations")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Solution: Implémentation de la Régression de Logistique en Python sur le Jeu de Donnée *Breast Cancer*

Solution : Implémentation de la Régression Logistique

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import seaborn as sns
sns.set_theme()

# --- Charger les données ---
data = load_breast_cancer()
X_raw = data.data
y = data.target
feature_names = data.feature_names

# --- Division train / validation AVANT standardisation ---
X_raw_train, X_raw_val, y_train, y_val = train_test_split(
    X_raw, y, test_size=0.2, random_state=8302
)

# --- Standardisation ---
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_raw_train)
X_val_std = scaler.transform(X_raw_val)

# --- Ajout de l'intercept ---
X_train = np.column_stack((np.ones(X_train_std.shape[0]), X_train_std))
X_val = np.column_stack((np.ones(X_val_std.shape[0]), X_val_std))
variables = ['Intercept'] + list(feature_names)
```

Solution : Implémentation de la Régression Logistique

```python
# --- Fonctions log-vraisemblance et gradient ---

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def log_likelihood_logistic(beta, X, y):
    eta = X @ beta
    eta = np.clip(eta, -30, 30)
    p = sigmoid(eta)
    return np.sum(y * np.log(p + 1e-12) + (1 - y) * np.log(1 - p + 1e-12))

def gradient_logistic(beta, X, y):
    eta = X @ beta
    eta = np.clip(eta, -30, 30)
    p = sigmoid(eta)
    return X.T @ (y - p)

# --- Descente de gradient ---
def logistic_regression(X, y, lr=1e-4, max_iter=10000, tol=1e-6, X_val=None, y_val=None):
    beta = np.zeros(X.shape[1])
    ll_history_train = []
    ll_history_val = []
    for iteration in range(max_iter):
        grad = gradient_logistic(beta, X, y)
        beta = beta + lr * grad
        ll_train = log_likelihood_logistic(beta, X, y)
        ll_history_train.append(ll_train)
        if X_val is not None and y_val is not None:
            ll_val = log_likelihood_logistic(beta, X_val, y_val)
            ll_history_val.append(ll_val)
        if iteration > 0 and abs(ll_history_train[-1] - ll_history_train[-2]) < tol:
            print(f"Convergence atteinte en {iteration} itérations.")
            break
    return beta, ll_history_train, ll_history_val
```

Solution : Implémentation de la Régression Logistique

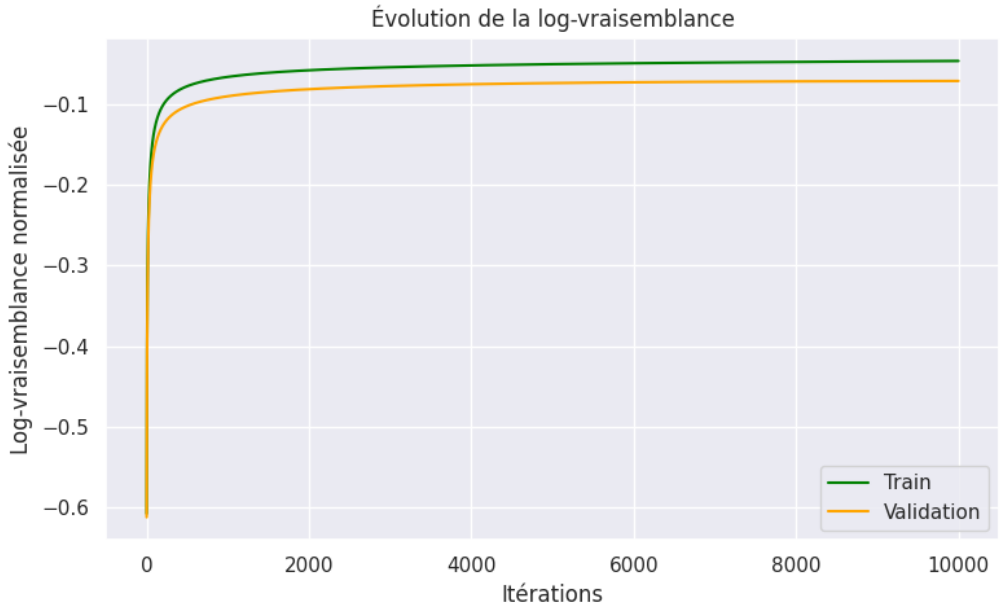
```python
# --- Entraînement ---
beta_hat, history_train, history_val = logistic_regression(
    X_train, y_train, X_val=X_val, y_val=y_val
)

# --- Affichage des coefficients estimés ---
print("Coefficients estimés :")
for var, coef in zip(variables, beta_hat):
    print(f"  {var} : {coef:.4f}")

# --- Visualisation : Log-vraisemblance ---
plt.figure(figsize=(8, 5))
plt.plot(np.array(history_train)/len(y_train), label="Train", color="green")
plt.plot(np.array(history_val)/len(y_val), label="Validation", color="orange")
plt.xlabel("Itérations")
plt.ylabel("Log-vraisemblance normalisée")
plt.title("Évolution de la log-vraisemblance")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# --- Visualisation : Prédictions vs Observations ---
p_pred = sigmoid(X_val @ beta_hat)

y_pred_class = (p_pred >= 0.5).astype(int) # Seuil de décision est p = 0.5.
plt.figure(figsize=(8, 5))
plt.scatter(p_pred, y_val, alpha=0.6, color='royalblue')
plt.axvline(x=0.5, color='red', linestyle='--', linewidth=1.5, label='Seuil de décision (0.5)')
plt.xlabel("Probabilité prédite")
plt.ylabel("Classe observée")
plt.title("Régression Logistique | Prédictions vs Observations")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Résultats — Régression Logistique

```
Coefficients estimés :
  Intercept : 0.1159
  mean radius : -0.6371
  mean texture : -0.4722
  mean perimeter : -0.5736
  mean area : -0.7119
  mean smoothness : -0.4782
  mean compactness : 0.7346
  mean concavity : -0.7749
  mean concave points : -1.0520
  mean symmetry : 0.2526
  mean fractal dimension : 0.4211
  radius error : -1.5648
  texture error : -0.0687
  perimeter error : -1.0656
  area error : -1.1429
  smoothness error : -0.2516
  compactness error : 0.9291
  concavity error : -0.0327
  concave points error : -0.4268
  symmetry error : 0.3495
  fractal dimension error : 0.9970
  worst radius : -1.2277
  worst texture : -1.2949
  worst perimeter : -1.0594
  worst area : -1.1812
  worst smoothness : -1.0218
  worst compactness : 0.1065
  worst concavity : -1.0446
  worst concave points : -0.9288
  worst symmetry : -1.2456
  worst fractal dimension : -0.5442
```

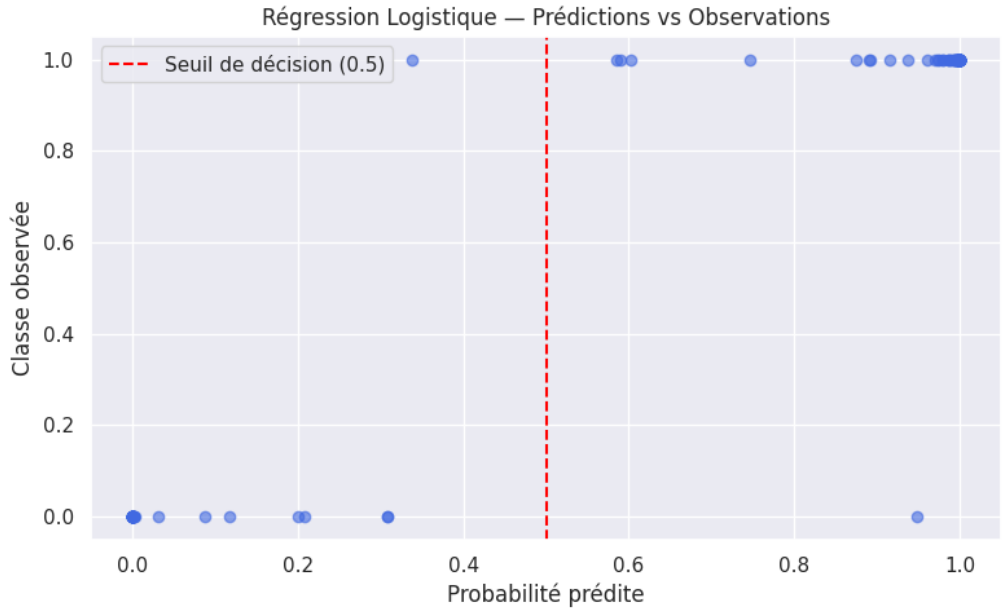
Interprétation des Résultats

- **Interprétation de la log-vraisemblance :**
  - Le graphe montre une **convergence stable** des log-vraisemblances en train et validation.
  - **Pas de sur-apprentissage visible** : la courbe de validation reste proche de celle du train.
  - Le modèle semble bien entraîné, atteignant un **plateau sans oscillation ni divergence**.
  - Le modèle **généralise bien** aux données de validation (pas d'overfitting détecté).



Interprétation des Résultats

- **Interprétation du graphique Prédictions vs Observations :**
  - Pour $y = 1$ (positif) : des probabilités proches de 1 $\Rightarrow$ **bon pouvoir discriminant**.
  - Pour $y = 0$: des probabilités proches de 0 $\Rightarrow$ le modèle **distingue bien les classes**.
  - Quelques points intermédiaires (proches de 0.5) suggèrent de **légères zones d'incertitude**.
  - Globalement, la séparation est **nette** et le **pouvoir prédictif** du modèle est bon.



Métriques d'Évalutation et Outils de Visualisation

Métriques d'Évalutation et Outils de Visualisation

- Une fois notre modèle de régression logistique entraîné, il faudra l'évaluer.
- Voici quelques métriques et outils de visualisation couramment utilisées pour évaluer un modèle de classification binaire :
  - **Accuracy (exactitude) :** Proportion de prédictions correctes (positifs + négatifs).
  - **Précision :** Parmi les observations prédites comme positives, combien sont réellement positives ?
  - **Rappel (recall) :** Parmi les vraies positives, combien ont été correctement détectées ?
  - **F1-score :** Moyenne harmonique entre précision et rappel — utile en cas de déséquilibre.
  - **Courbe ROC et AUC :** Évalue le comportement du classifieur à tous les seuils de décision.
  - **Matrice de confusion :** Résume les performances du modèle pour chaque classe prédite vs. réelle.
  - **Frontière de décision :** Représente visuellement la séparation entre les classes selon le modèle.

Métriques d'Évalutation et Outils de Visualisation : Exactitude (Accuracy)

Métriques d'Évalutation et Outils de Visualisation : Acc

- **Exactitude (Accuracy) Définition :**

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total d'observations}}$$
$$= \frac{\text{VP (Vrais Pos)} + \text{VN (Vrais Nég)}}{\text{VP (Vrais Pos)} + \text{FN (Vrais Nég)} + \text{FP (Faux Pos)} + \text{FN (Faux Nég)}}.$$

- **Significations de VP, VN, FP et FN:**
  - VP (Vrai Positif) : Le modèle a prédit 1, et c'était bien 1.
  - VN (Vrai Négatif) : Le modèle a prédit 0, et c'était bien 0.
  - FP (Faux Positif) : Le modèle a prédit 1, mais c'était 0.
  - FN (Faux Négatif) : Le modèle a prédit 0, mais c'était 1.
- **Intuition :**
  - C'est la proportion d'observations pour lesquelles le modèle a prédit correctement la classe.
  - **Exemple :** Si on a 90 bonnes prédictions sur 100 observations, l'accuracy est de 90%.
- **Utilité :**
  - Simple à comprendre et à calculer.
  - Donne une vue d'ensemble de la performance du modèle.

Métriques d'Évalutation et Outils de Visualisation : Acc

- **Exactitude (Accuracy) Définition :**

$$\text{Accuracy} = \frac{\text{Nombre de bonnes prédictions}}{\text{Nombre total d'observations}}$$
$$= \frac{\text{VP (Vrais Pos)} + \text{VN (Vrais Nég)}}{\text{VP (Vrais Pos)} + \text{FN (Vrais Nég)} + \text{FP (Faux Pos)} + \text{FN (Faux Nég)}}.$$

- **Intuition :**
  - C'est la proportion d'observations pour lesquelles le modèle a prédit correctement la classe.
  - **Exemple :** Si on a 90 bonnes prédictions sur 100 observations, l'accuracy est de 90%.
- **Utilité :**
  - Simple à comprendre et à calculer.
  - Donne une vue d'ensemble de la performance du modèle.
- **Limites :**
  - L'accuracy peut être trompeuse si les classes sont déséquilibrées.
  - Exemple : Si 95% des exemples sont de classe 0, un modèle qui prédit toujours 0 aura 95% d'accuracy, mais ne détecte jamais la classe 1!

Métriques d'Évalutation et Outils de Visualisation : Précision

Métriques d'Évalutation et Outils de Visualisation : Prec

- **Définition de la Précision:**

$$\text{Précision} = \frac{\text{VP (Vrais Positifs)}}{\text{VP (Vrais Positifs)} + \text{FP (Faux Positifs)}}$$

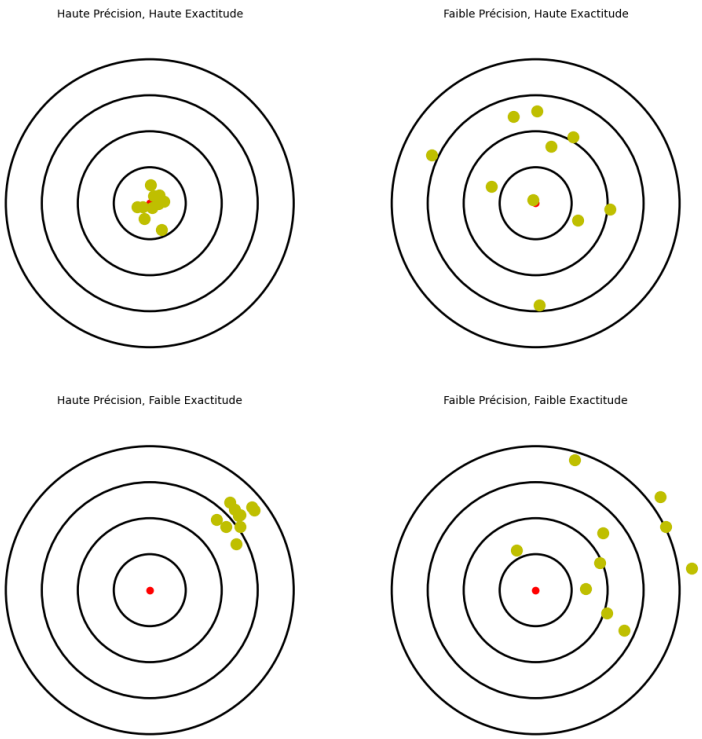
- **Intuition :**
  - La précision mesure la qualité des prédictions positives du modèle.
  - Autrement dit : **Quand le modèle prédit "positif", a-t-il raison?**
  - **Exemple :** Le modèle prédit que 100 personnes sont malades, mais seulement 80 le sont vraiment $\Rightarrow$ précision $= 80\%$.
- **Utilité :**
  - La précision est importante quand **faire une fausse alerte peut être un problème.**
  - **Exemple :** Dans un filtre anti-spam, on ne veut pas que des emails importants soient envoyés à la corbeille par erreur (faux positifs).
- **Limites :**
  - Un modèle peut avoir une très bonne précision simplement en ne prédisant "positif" que dans les cas très sûrs.
  - Il peut alors rater plein de vrais positifs $\Rightarrow$ il faut regarder aussi le **rappel**.

Métriques d'Évalutation et Outils de Visualisation

- Cette figure illustre deux notions essentielles en évaluation de modèles :
  - **la précision :** Relative à la dispersion des prédictions et donc à la variance du modèle.
  - **l'exactitude ou encore la justesse :** Relative à proximité par rapport à la vraie cible (vérité) et donc au biais du modèle.

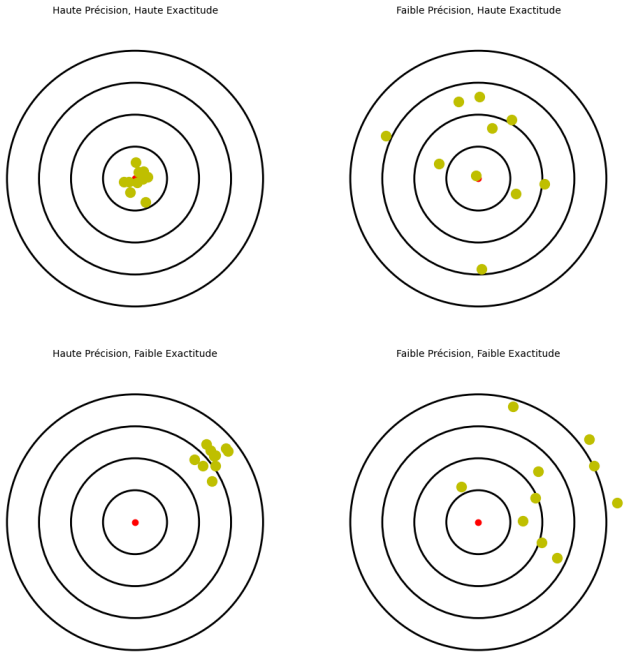
Métriques d'Évalutation et Outils de Visualisation

- Cette figure illustre deux notions essentielles en évaluation de modèles :
  - **Précision** : les prédictions sont-elles proches les unes des autres ?
  - **Exactitude (ou Justesse)** : les prédictions sont-elles proches de la vérité ?
- Ces notions sont illustrées par des fléchettes tirées sur une cible.



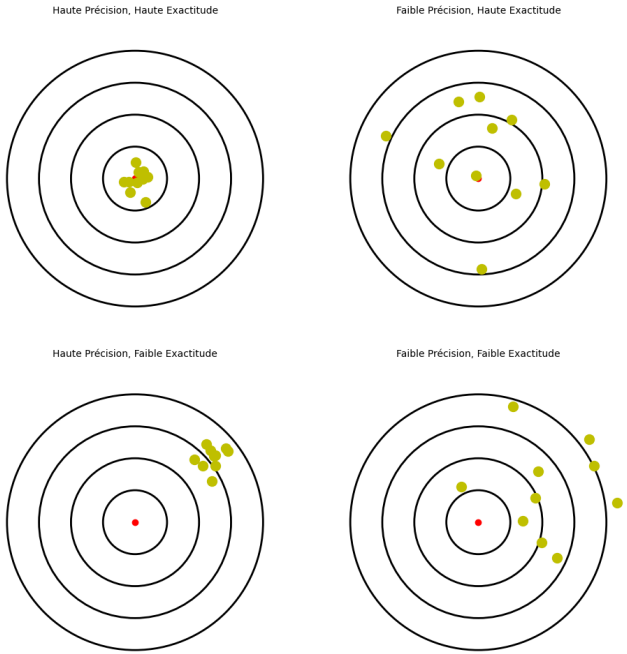
Métriques d'Évalutation et Outils de Visualisation

- **Grande précision, haute exactitude :**
  - Les fléchettes sont toutes proches du centre (la vérité) et très regroupées.
  - Le modèle est à la fois **précis** (variance faible) et **juste** (biais faible) et donc, proche de la vérité.
  - C'est ce qu'on vise idéalement : un modèle cohérent (précis) et performant (exacte).



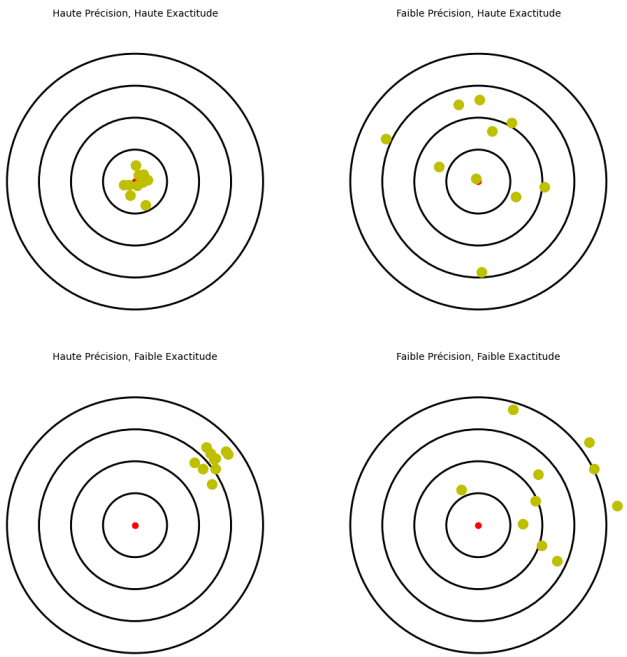
Métriques d'Évalutation et Outils de Visualisation

- **Faible précision, haute exactitude :**
  - Les fléchettes sont dispersées (grande variance), mais en moyenne proches du centre (juste en moyenne).
  - Le modèle donne de bonnes prédictions globalement, mais manque de stabilité (biais faible mais grande variance du modèle).
  - On dit qu'il est **juste**, mais pas **précis**.



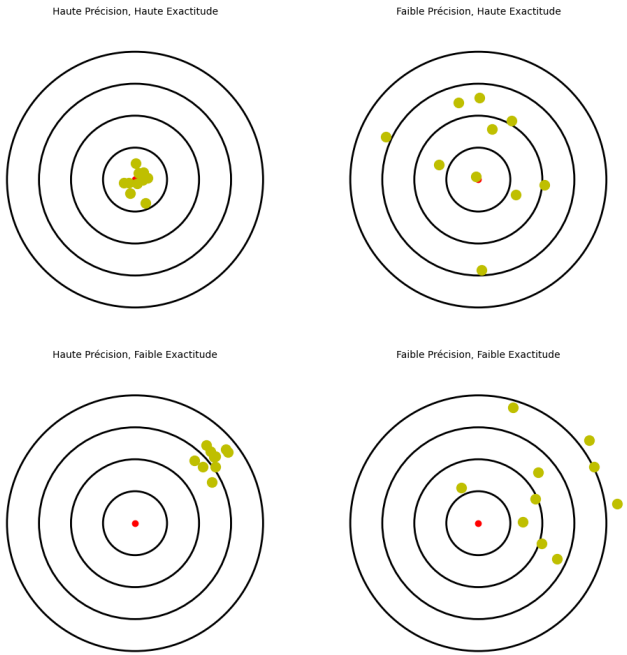
Métriques d'Évalutation et Outils de Visualisation

- **Grande précision, Faible exactitude :**
  - Les fléchettes sont très regroupées, mais loin du centre.
  - Le modèle est **cohérent** (variance faible), mais se trompe systématiquement (biais fort).
  - C'est souvent le cas d'un **modèle biaisé**.



Métriques d'Évalutation et Outils de Visualisation

- Les fléchettes sont dispersées et loin du centre.
- Le modèle est à la fois **instable** et **peu fiable**.
- C'est un mauvais modèle : **il manque de précision et d'exactitude (variance forte et biais fort).**



Métriques d'Évalutation et Outils de Visualisation : Acc

- **Exactitude (Accuracy) Définition :**

$$\text{Accuracy} = \frac{\text{Nombre de bonnes prédictions}}{\text{Nombre total d'observations}}$$
$$= \frac{\text{VP (Vrais Pos)} + \text{VN (Vrais Nég)}}{\text{VP (Vrais Pos)} + \text{FN (Vrais Nég)} + \text{FP (Faux Pos)} + \text{FN (Faux Nég)}}.$$

- **Intuition :**
  - C'est la proportion d'observations pour lesquelles le modèle a prédit correctement la classe.
  - **Exemple :** Si on a 90 bonnes prédictions sur 100 observations, l'accuracy est de 90%.
- **Utilité :**
  - Simple à comprendre et à calculer.
  - Donne une vue d'ensemble de la performance du modèle.
- **Limites :**
  - L'accuracy peut être trompeuse si les classes sont déséquilibrées.
  - Exemple : Si 95% des exemples sont de classe 0, un modèle qui prédit toujours 0 aura 95% d'accuracy, mais ne détecte jamais la classe 1!

Métriques d'Évalutation et Outils de Visualisation : Prec

- **Définition de la Précision:**

$$\text{Précision} = \frac{\text{VP (Vrais Positifs)}}{\text{VP (Vrais Positifs)} + \text{FP (Faux Positifs)}}$$

- **Intuition :**
  - La précision mesure la qualité des prédictions positives du modèle.
  - Autrement dit : **Quand le modèle prédit "positif", a-t-il raison?**
  - **Exemple :** Le modèle prédit que 100 personnes sont malades, mais seulement 80 le sont vraiment $\Rightarrow$ précision $= 80\%$.
- **Utilité :**
  - La précision est importante quand **faire une fausse alerte peut être un problème.**
  - **Exemple :** Dans un filtre anti-spam, on ne veut pas que des emails importants soient envoyés à la corbeille par erreur (faux positifs).
- **Limites :**
  - Un modèle peut avoir une très bonne précision simplement en ne prédisant "positif" que dans les cas très sûrs.
  - Il peut alors rater plein de vrais positifs $\Rightarrow$ il faut regarder aussi le **rappel**.

Métriques d'Évalutation et Outils de Visualisation : Le Rappel

Métriques d'Évalutation et Outils de Visualisation : Rapp

- **Définition du Rappel:**

$$\text{Rappel} = \frac{\text{VP (Vrais Positifs)}}{\text{VP (Vrais Positifs)} + \text{FN (Faux Négatifs)}}$$

- **Intuition :**
  - Le rappel est un ratio qui mesure la capacité du modèle à **retrouver tous les cas positifs**.
  - Exemple : s'il y a 100 vraies personnes malades et que le modèle en détecte 80, alors le rappel est de 80%.
- **Utilité :**
  - Le rappel est important quand il est **grave d'oublier un cas positif**.
  - **Exemple :** En médecine, il vaut mieux détecter tous les patients malades.
- **Limites :**
  - Un modèle peut avoir un bon rappel mais faire beaucoup d'erreurs (précision faible).
  - Il faut donc regarder aussi d'autres métriques pour évaluer la qualité globale du modèle comme le F1-score.

Métriques d'Évalutation et Outils de Visualisation : F1-score

Métriques d'Évalutation et Outils de Visualisation : F1

- **Définition du F1-score:**

$$\text{F1-score} = 2 \cdot \frac{\text{Précision} \cdot \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

- **Intuition :**
  - Le F1-score combine la **précision** et le **rappel** en une seule mesure.
  - Il ne sera élevé que si les deux sont élevés.
  - Il donne un bon score uniquement quand le modèle est à la fois bon pour **trouver les vrais positifs** et pour **éviter les faux positifs.**
- **Utilité :**
  - Le F1-score est très utile quand on cherche un équilibre entre **ne pas rater de vrais cas** (rappel) et **ne pas se tromper sur les cas prédits positifs** (précision).
  - **Exemple :** En détection de maladies, on veut détecter un maximum de malades sans faussement alarmer les personnes saines.
- **À retenir :** Si le rappel ou la précision sont faibles, le F1-score sera faible aussi. **Il permet de mieux évaluer les modèles dans des situations où l'accuracy (exactitude) est trompeuse ou bien les classes sont déséquilibrées.**

Métriques d'Évalutation et Outils de Visualisation

- **Seuil de Décision** (paramètre Bernoulli) :
  - Pour un modèle comme la régression logistique qui prédit une **probabilité** (ex. $p = 0.5$), pour prédire une classe (0 ou 1), on utilise un **seuil de décision** :

$$\hat{y} = \begin{cases} 1 & \text{si } p \geq \text{seuil} \\ 0 & \text{sinon} \end{cases}$$

  - Le seuil par défaut est 0.5, mais ce **n'est pas toujours optimal**.
- **Pourquoi ajuster le seuil est important?**
  - Baisser le seuil $\Rightarrow$ Favorise le **rappel** (augmente la confiance du modèle à considérer des positifs, et par conséquent, à **détecter plus de vrais positifs).**
  - Augmenter le seuil $\Rightarrow$ Favorise la **précision** (pousse le modèle à considérer des positifs que s'il est très confiant, et par conséquent, **favorise moins de faux positifs).**
- **Exemple concret en médecine :** On veut être sûr de ne pas rater un patient malade $\Rightarrow$ on baisse le seuil (ex. 0.3). On veut éviter d'inquiéter inutilement des personnes saines $\Rightarrow$ on monte le seuil (ex. 0.7).

Métriques d'Évalutation et Outils de Visualisation : Matrice de Confusion

Métriques d'Évalutation et Outils de Visualisation : MC

- **Définition de la Matrice de Confusion:**
  - La matrice de confusion est un tableau qui résume les résultats de classification d'un modèle binaire.
  - Elle compare les classes réelles aux classes prédites par le modèle.
  - Elle est à la base du calcul de nombreuses métriques comme la précision, le rappel et le F1-score.

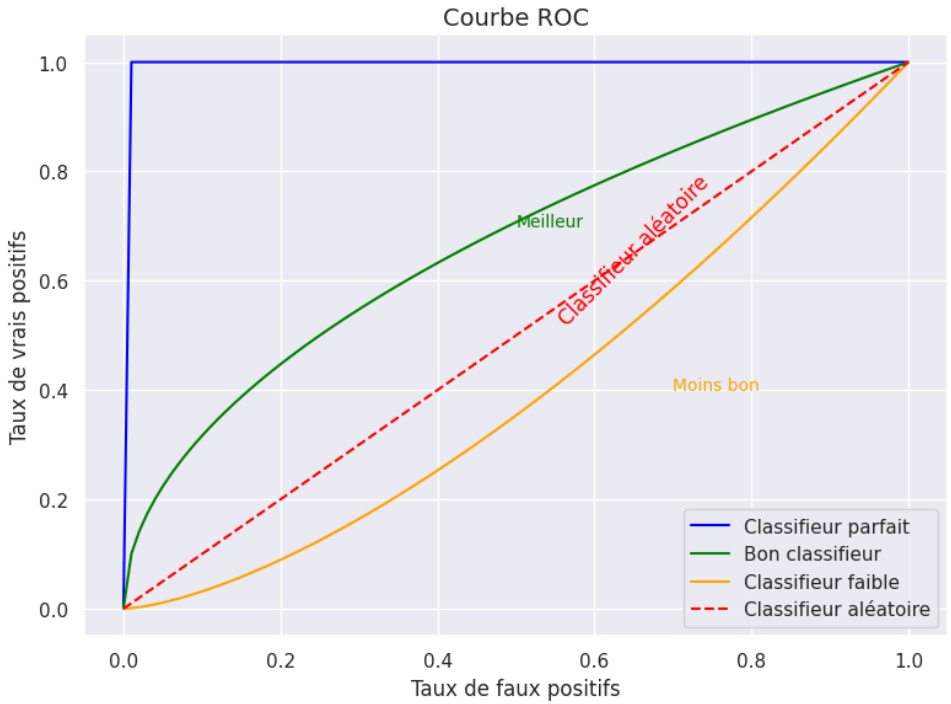
	Prédit : 0	**Prédit : 1**
Réel : 0	VN (True Negative)	FP (Faux Positif)
Réel : 1	FN (Faux Négatif)	VP (Vrai Positif)

- **Intuition :**
  - Elle permet de visualiser **où le modèle se trompe :** confond-il les classes? A-t-il tendance à surestimer ou sous-estimer les positifs?
- **Utilité :**
  - Elle offre une vue détaillée des erreurs du modèle.
  - Permet d'adapter son seuil ou sa stratégie selon les erreurs qu'on veut éviter (faux positifs ou faux négatifs).

Métriques d'Évalutation et Outils de Visualisation : Courbe ROC

Métriques d'Évalutation et Outils de Visualisation : ROC

- **Courbe ROC (Receiver Operating Characteristic) :** La courbe ROC trace pour **tous les seuils possibles :**
  - Axe des y : Taux de vrais positifs (TVP, c'est le Rappel)
  - Axe des x : Taux de faux positifs (TFP)
  - Chaque point de la courbe correspond à un seuil de décision différent.

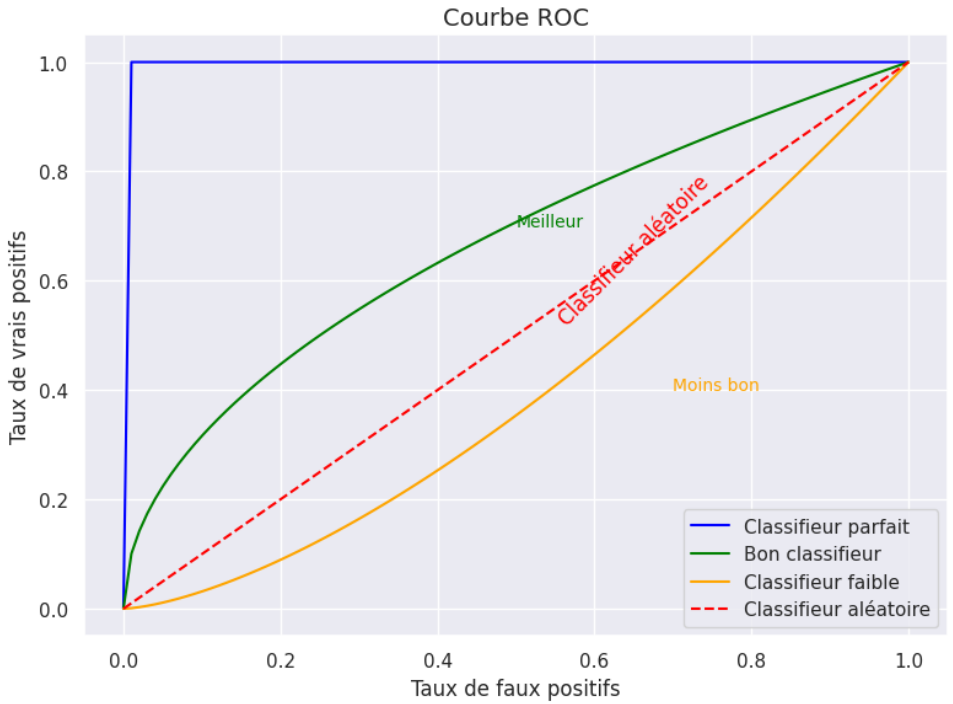


Définitions Formelles : TPR et FPR

True Positive Rate (TPR) $\quad TVP = \dfrac{TP}{TP + FN}$ (C'est le **Rappel** (Recall))

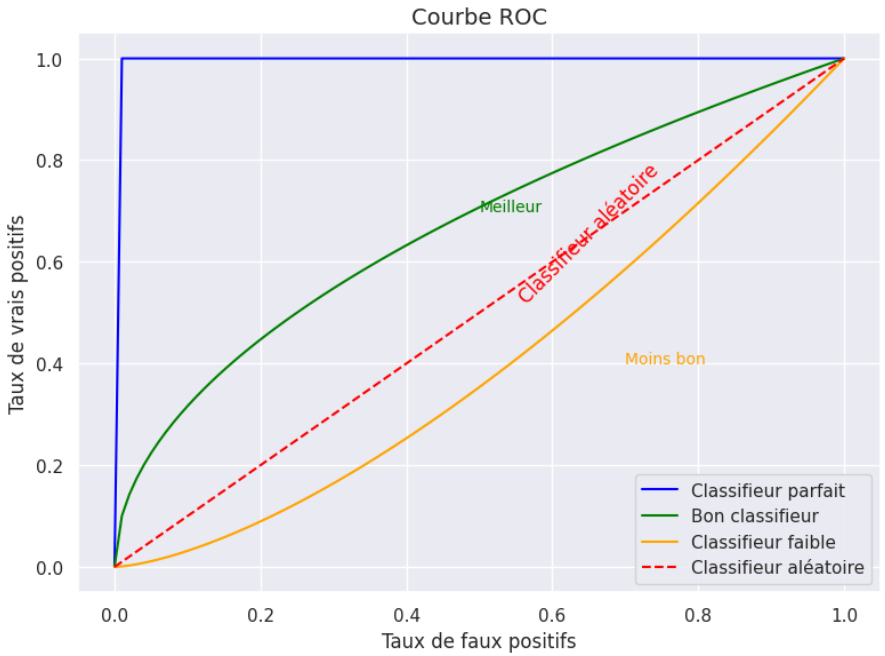
False Positive Rate (FPR) $\quad TFP = \dfrac{FP}{FP + TN}$

Table: Formules utilisées pour tracer la courbe ROC



Intuition derrière la courbe ROC (Receiver Operating Characteristic)

- Le classifieur binaire produit probabilité $\hat{p}$.
- Pour prédire la classe (0 ou 1), on fixe un seuil p (souvent $= 0.5$).
- Comment le modèle se comporte-t-il pour tous les seuils?
- $\Rightarrow$ La courbe ROC résume les performances du modèle pour tous les seuils. Elle illustre le compromis entre la détection des vrais positifs (TVP) et les erreurs pour les négatifs (TFP).

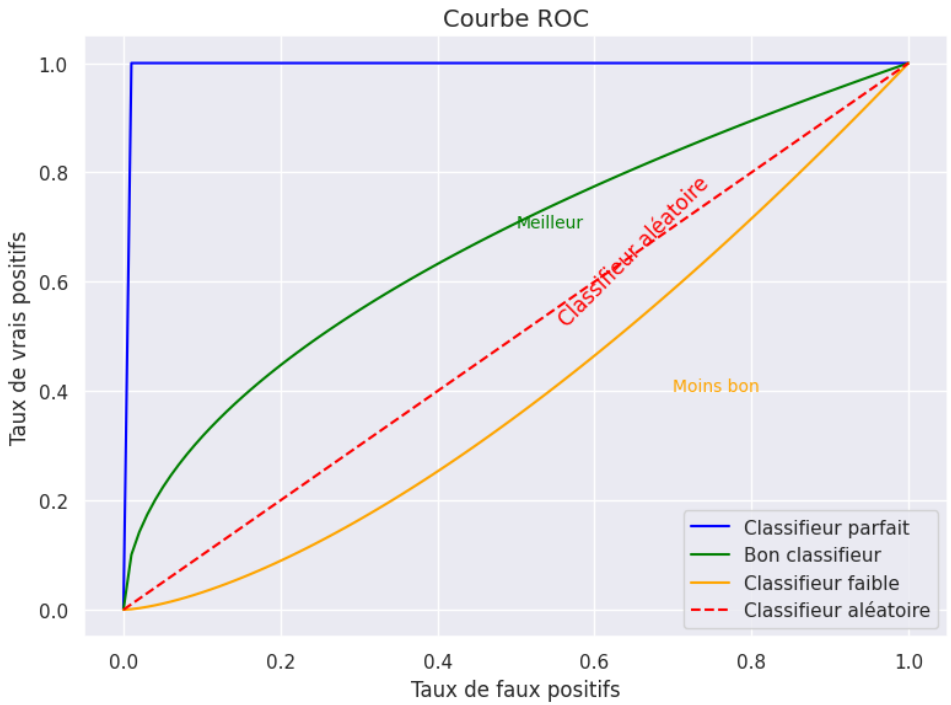


Comment se construit la courbe ROC ?

- Pour chaque seuil $p \in [0, 1]$:
  - Transformer les probabilités en classes : 1 si $\hat{p} \geq p$, sinon 0.
  - Calculer :
    - $\text{TVP} = \frac{\text{TP}}{\text{TP+FN}}$ = proportion des vrais positifs détectés.
    - $\text{TFP} = \frac{\text{FP}}{\text{FP+TN}}$ proportion des faux positifs parmi les négatifs.
  - Chaque seuil p produit un point (TFP, TVP) sur le graphique.
- En faisant varier le seuil p de 0 à 1, on trace toute la courbe.

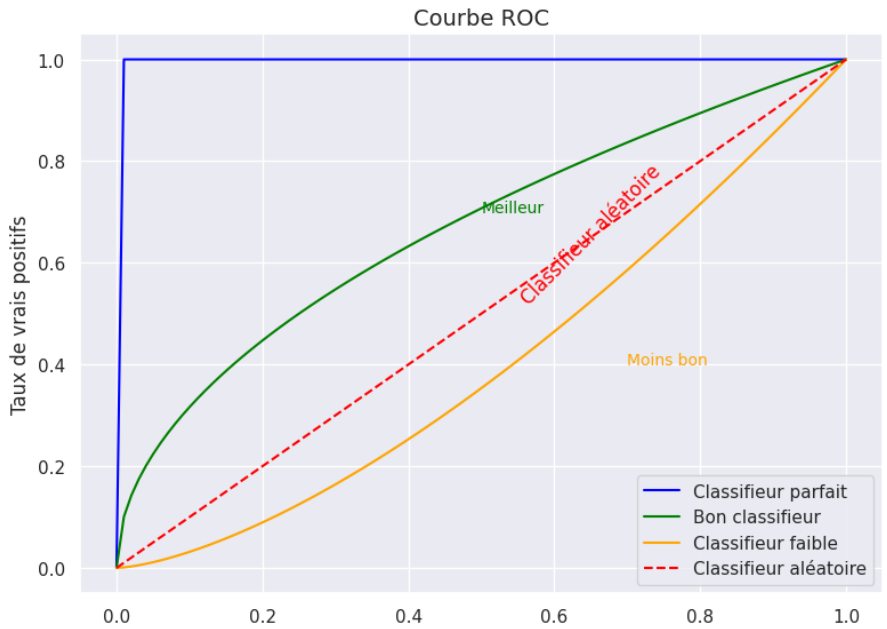
Axes de la courbe ROC

- Axe des abscisses (X) :
  - $\text{TFP} = \frac{\text{FP}}{\text{FP+TN}}$
  - **Correspond aux erreurs sur les exemples de classe 0.**
- Axe des ordonnées (Y) :
  - $\text{TVP} = \frac{\text{TP}}{\text{TP+FN}}$
  - **Correspond aux exemples de classe 1 correctement détectés.**



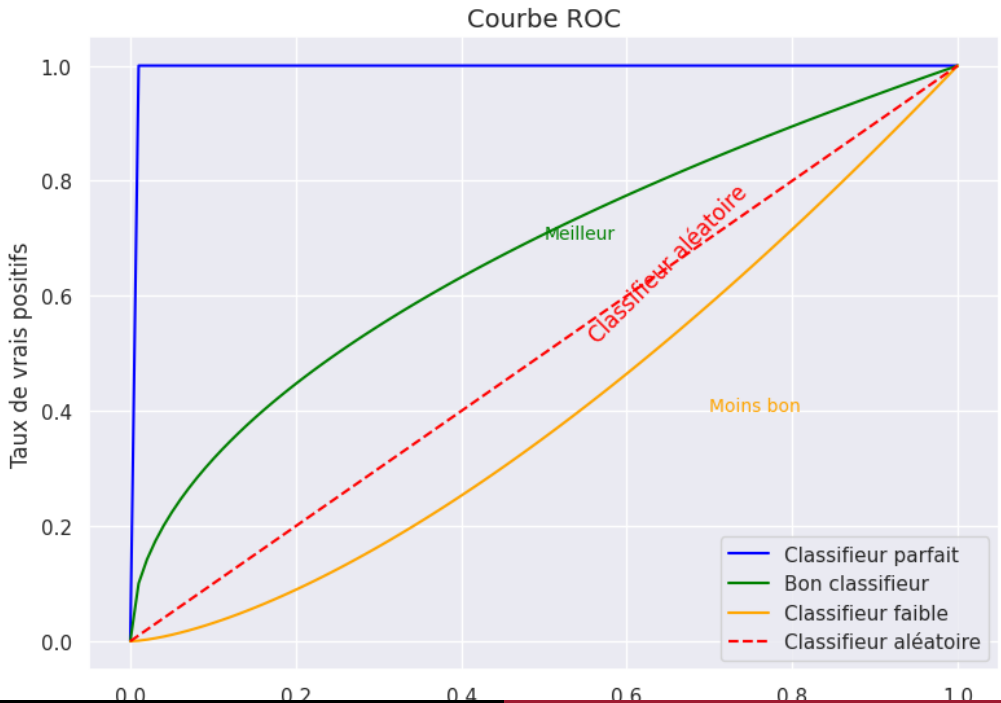
Métriques d'Évalutation et Outils de Visualisation : ROC

- **Intuition :**
  - On fait varier le seuil de 0 à 1, et on observe le **compromis entre correctement les exemples de la classe (1) positive (rappel ↑)** et **faire des d'erreurs sur les exemples négatifs (faux positifs ↑).**
  - Un modèle parfait monte à 1 immédiatement (aucune erreur), puis reste plat.
  - Un modèle aléatoire suit la diagonale : chaque gain de TPR correspond à une augmentation de FPR similaire.
  - Un modèle mauvais est sous la diagonale : il inverse les classes.



Métriques d'Évalutation et Outils de Visualisation : ROC

- **Interprétation des courbes :**
  - En bleu : un classifieur parfait (prédit correctement sans erreurs).
  - En vert : un classifieur efficace (bon compromis rappel/faux positifs).
  - En rouge pointillé : un classifieur aléatoire (tirage au sort). **Tout classifieur utile doit faire mieux.**
  - En orange : un classifieur médiocre (performance pire que le classifieur aléatoire). **Tout classifieur aléatoire ou utile doit faire mieux.**



Métriques d'Évalutation et Outils de Visualisation : ROC

- **Utilité :**
  - Permet d'évaluer **la performance globale du modèle** indépendamment d'un seuil fixe.
  - Utile pour comparer plusieurs modèles entre eux.
  - Permet de choisir un **seuil optimal** selon les besoins (précision vs rappel puisque le FPR est relatif à la précision) **lorsque les classes sont équilibrées.**

Métriques d'Évalutation et Outils de Visualisation : ROC

- **Choix di seuil : Tout dépend du contexte du problème :**
  - Dans des contextes critiques (ex. détection de fraude, diagnostic médical), on cherche à **maximiser le TPR** (ne rien rater), quitte à avoir plus de faux positifs.
  - Dans d'autres contextes (ex. détection de spam), on peut **tolérer des faux négatifs** pour réduire les faux positifs.
  - Il n'existe pas de **seuil optimal universel** : le choix dépend du **coût des erreurs** dans l'application.
- **Métriques d'Évalutation et Outils de Visualisation : ROC**
  - Ce coin correspond à : $\text{FPR} = 0$, $\text{TPR} = 1$
  - On cherche le point (FPR, TPR) qui minimise la distance à $(0, 1)$:

$$\text{Distance} = \sqrt{(1 - \text{TPR})^2 + (\text{FPR})^2}$$

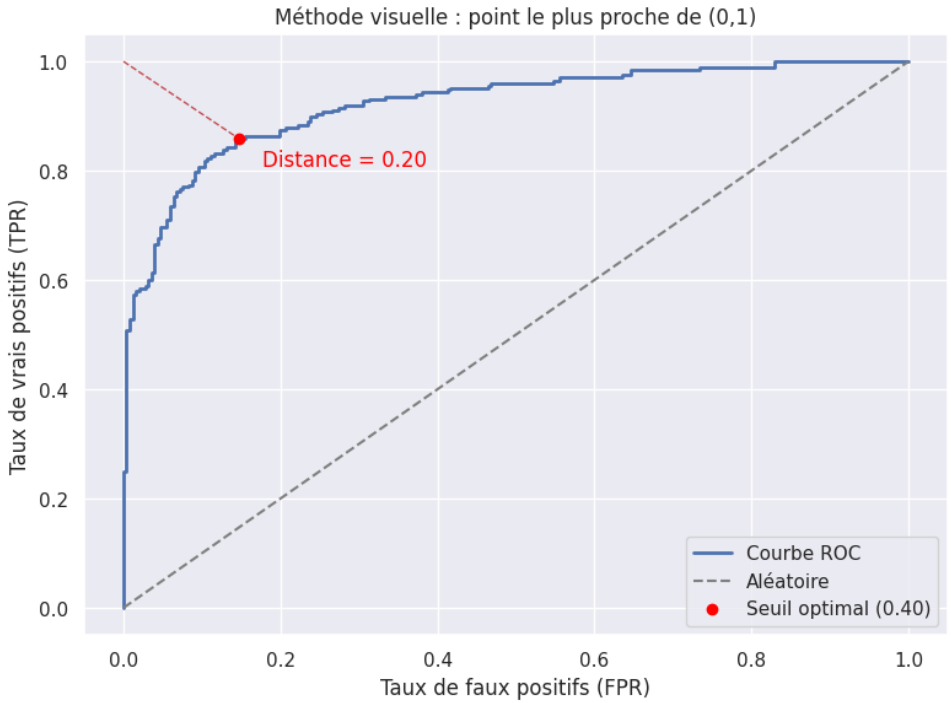
  - Ce seuil maximise le TPR tout en minimisant le FPR.

Métriques d'Évalutation et Outils de Visualisation : ROC

- **Méthode visuelle : choisir le point le plus proche du coin supérieur gauche**
  - Ce coin correspond à : $\text{FPR} = 0$, $\text{TPR} = 1$
  - On cherche le point (FPR, TPR) qui minimise la distance à $(0, 1)$:

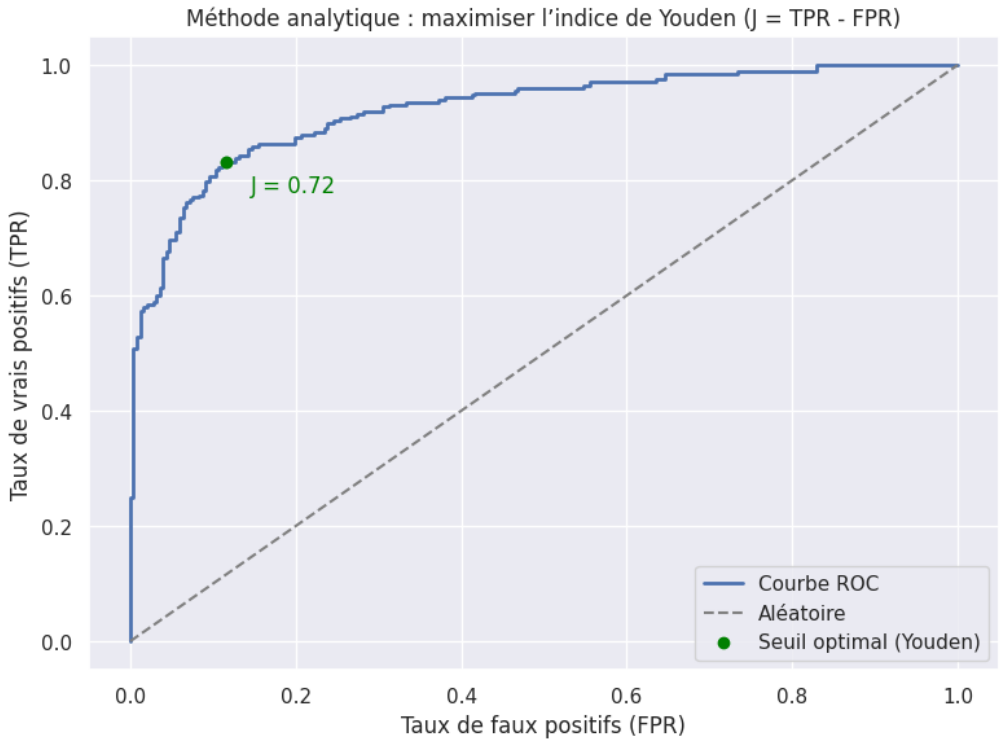
$$\text{Distance} = \sqrt{(1 - \text{TPR})^2 + (\text{FPR})^2}$$

  - Ce seuil maximise le TPR tout en minimisant le FPR.



Métriques d'Évalutation et Outils de Visualisation : ROC

- **Méthode analytique : maximiser l'indice de Youden (J)**
  - Définition : $J = \text{TPR} - \text{FPR}$
  - Il mesure le bénéfice net, en maximisant la bonne détection (TPR) et en limitant les erreurs (FPR).
  - Le seuil optimal est celui qui maximise J.

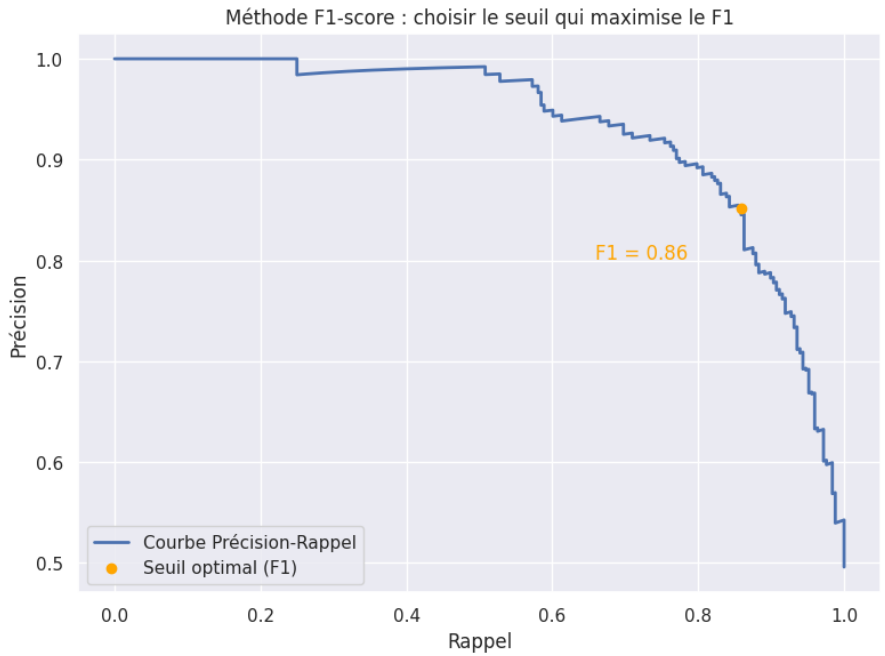


Métriques d'Évalutation et Outils de Visualisation : F1

- **Alternative - Maximiser le F1-score ou la précision à un seuil donné :** La courbe ROC n'est pas toujours pertinente, **surtout en cas de classes déséquilibrées.**
  - Dans ce cas, on peut :
    - Tracer une courbe précision-rappel (PR curve)
    - Calculer le F1-score pour chaque seuil :

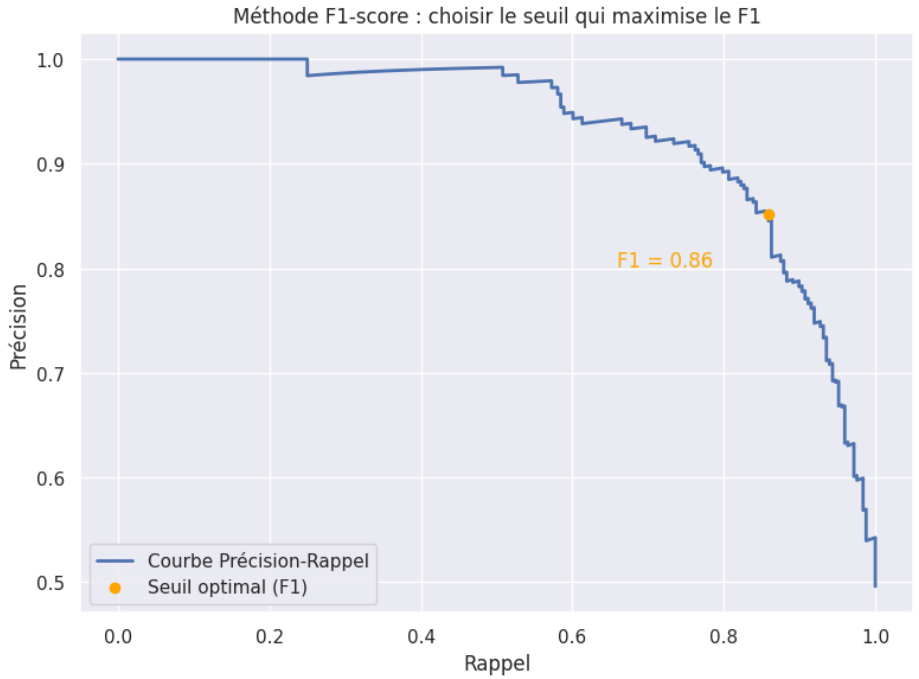
$$F_1 = 2 \cdot \frac{\text{Précision} \cdot \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

    - Choisir le seuil qui maximise F_1



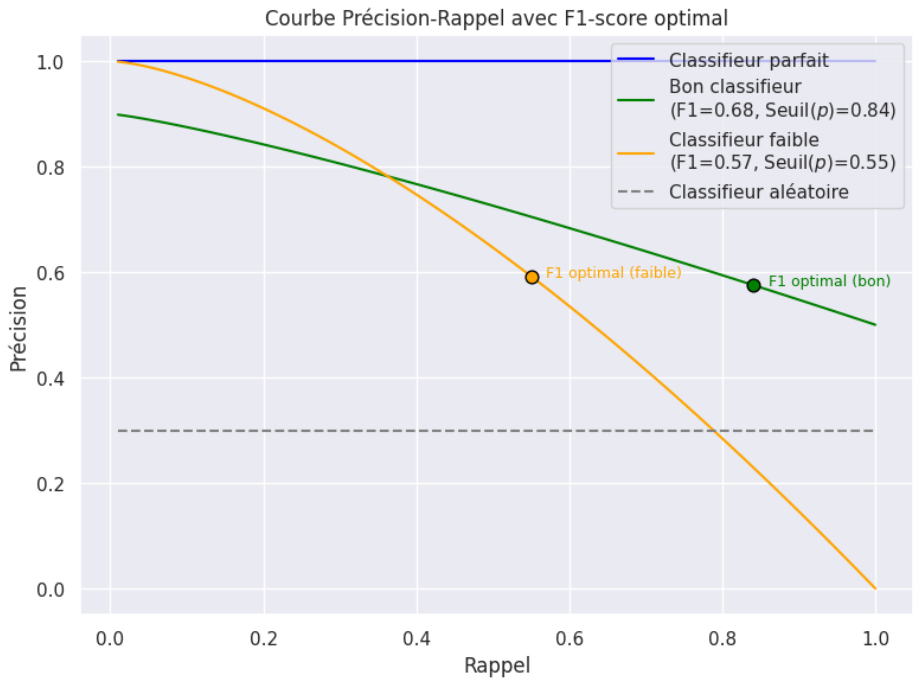
Métriques d'Évalutation et Outils de Visualisation : F1

- On fait varier le seuil de 0 à 1 :
  - Si le seuil est bas : rappel ↑, mais précision ↓ (plus de faux positifs).
  - Si le seuil est élevé : précision ↑, mais rappel ↓ (moins de vrais positifs détectés).
- **Compromis entre** la qualité des prédictions positives (précision) et la capacité à détecter tous les vrais positifs (rappel).



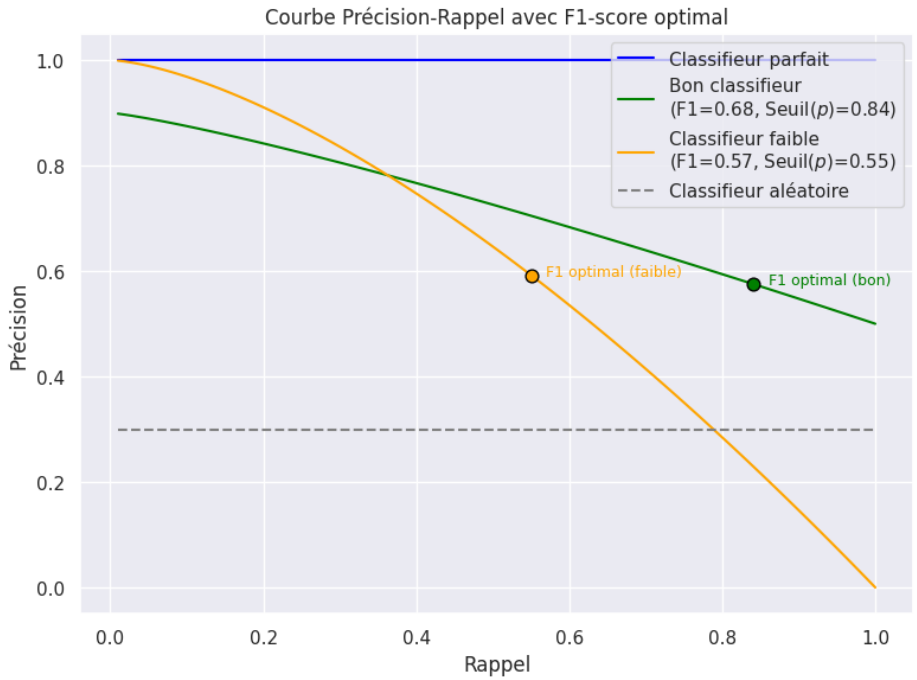
Métriques d'Évalutation et Outils de Visualisation : F1

- **Classifieur parfait**
  - Précision $= 1$ pour tout rappel.
  - Il classe tous les positifs correctement et ne produit aucun faux positif.
  - Courbe horizontale à 1 sur tout l'intervalle de rappel.
  - F1-score maximal $= 1$, peu importe le seuil.



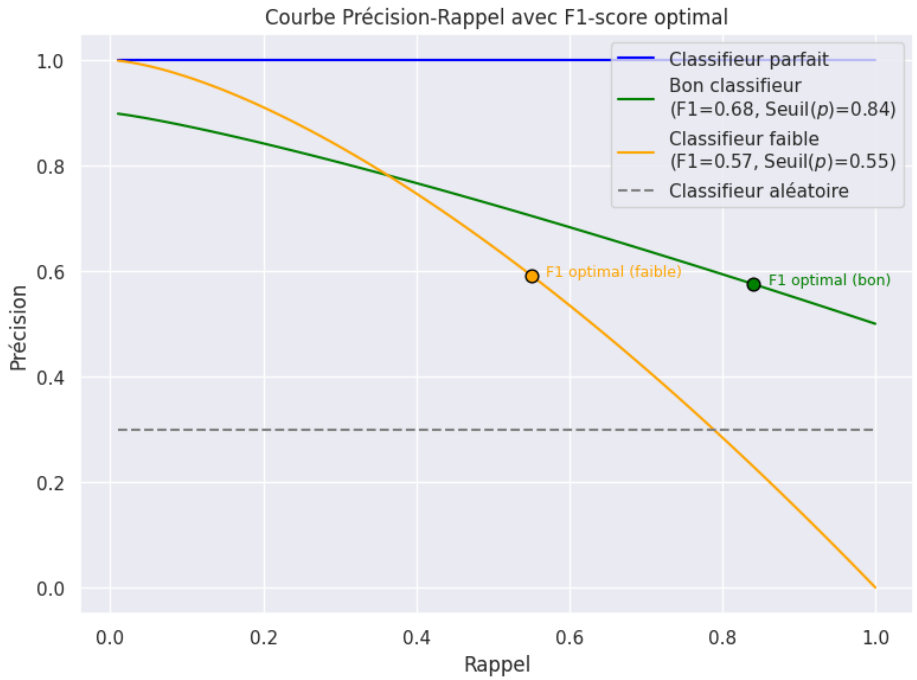
Comportement attendu des classifieurs dans la courbe Précision-Rappel

- **Bon classifieur**
  - Bonne précision initiale, qui diminue doucement lorsque le rappel augmente.
  - Il détecte bien les positifs avec peu de faux positifs.
  - Courbe concave et au-dessus de la courbe du modèle faible.
  - Le meilleur F1-score est souvent atteint pour un rappel élevé.



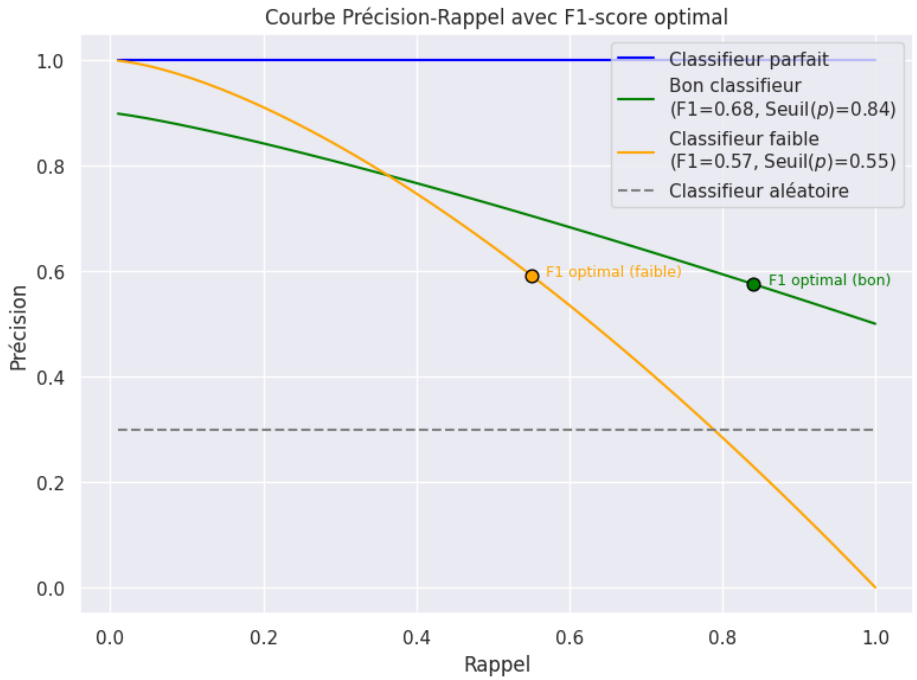
Comportement attendu des classifieurs dans la courbe Précision-Rappel

- **Classifieur faible**
  - Peut détecter certains positifs mais produit beaucoup de FP.
  - La précision chute rapidement dès qu'on cherche à augmenter le rappel : Courbe en forte décroissance.
  - F1-score optimal atteint à un compromis intermédiaire, mais reste inférieur à celui d'un bon classifieur.



Comportement attendu des classifieurs dans la courbe Précision-Rappel

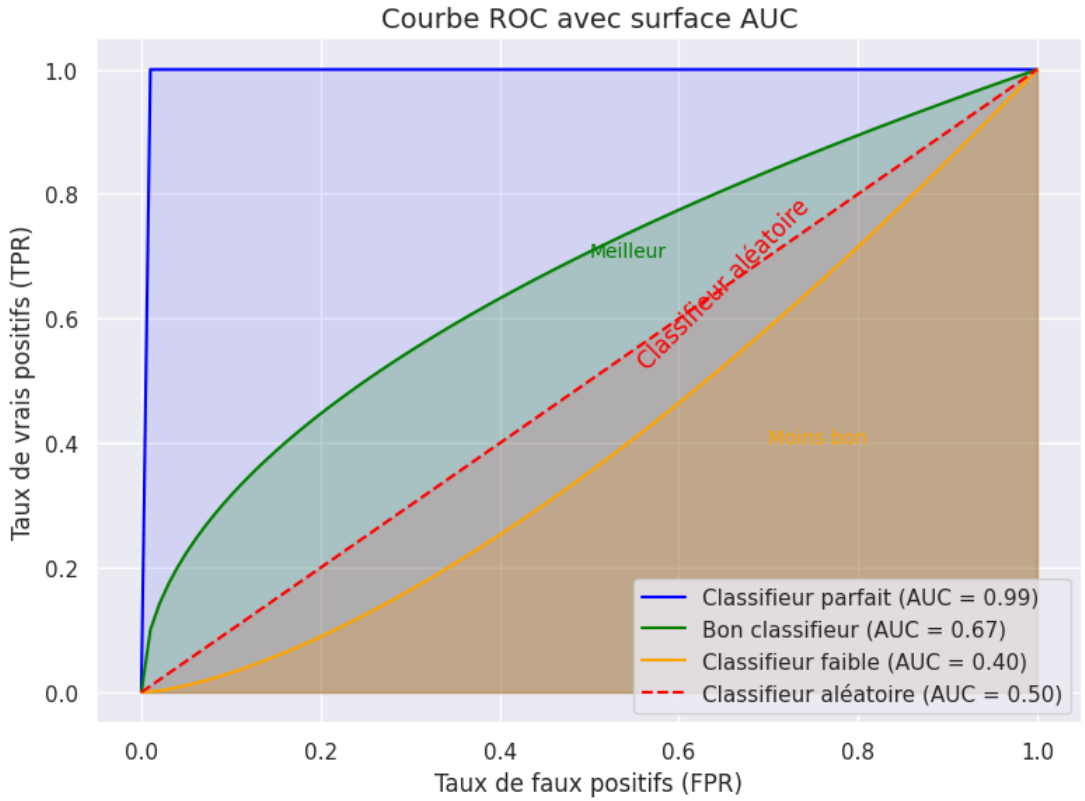
- **Classifieur aléatoire**
  - La précision est constante quelle que soit la valeur du rappel.
  - Précision égale à la proportion de la classe positive dans les données (ex. 30% ici).
  - Référence minimale (baseline) : tout classifieur utile doit faire mieux.
  - F1-score faible et constant, sans réel point optimal.



Métriques d'Évalutation et Outils de Visualisation : AUC (Area Under the Curve)

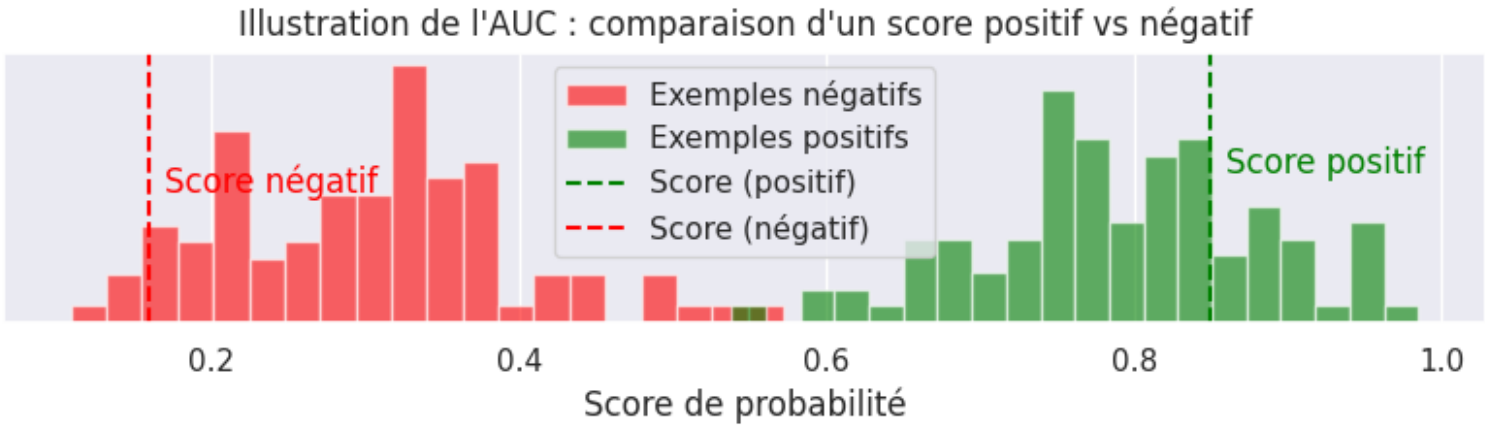
Métriques d'Évalutation et Outils de Visualisation : AUC

- **AUC (Area Under the Curve) :**
  - L'AUC correspond à l'**aire sous la courbe ROC**.
  - Sa valeur est comprise entre 0 et 1 :
    - AUC = 1 : modèle parfait.
    - AUC = 0.5 : modèle aléatoire (aucune capacité à distinguer les classes).



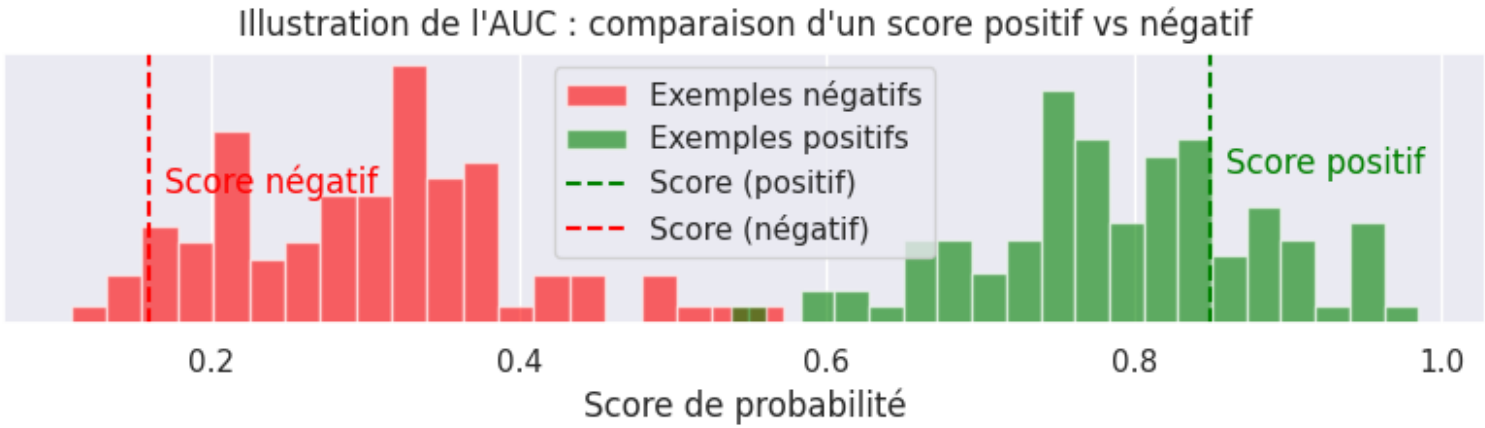
Métriques d'Évalutation et Outils de Visualisation : AUC

- Le modèle attribue des scores (probabilités) à chaque observation.
- L'AUC mesure **si, en moyenne**, les scores des positifs sont plus élevés que ceux des négatifs.
- Si le modèle donne toujours un score plus élevé aux vrais positifs qu'aux vrais négatifs, alors **l'AUC est proche de 1**.
- Si le modèle se trompe souvent dans cet ordre, alors **l'AUC est proche de 0.5** (comme un tirage au sort).
- C'est une manière de mesurer le **pouvoir de classement** du modèle, pas seulement sa capacité à prédire correctement à un seuil donné (comme 0.5).



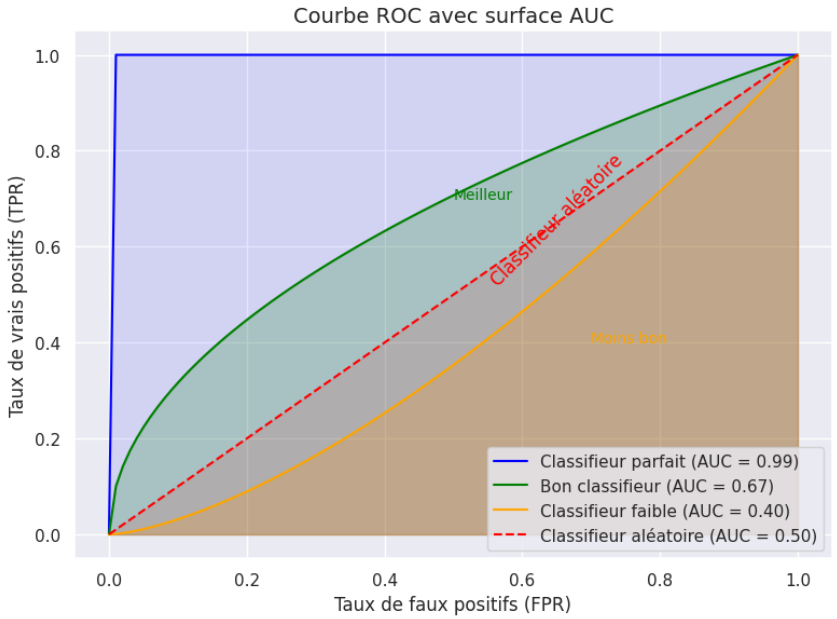
Métriques d'Évalutation et Outils de Visualisation : AUC

- On compare deux distributions de scores : une pour les exemples positifs, l'autre pour les exemples négatifs.
- Dans la figure ci-dessous, on a tiré un score au hasard dans chaque distribution.
- L'exemple **positif** a un score **plus élevé** que l'exemple négatif.
- L'AUC correspond à la probabilité qu'une telle situation se produise.
- C'est une mesure **indépendante du seuil** qui capture la **capacité de discrimination globale** du modèle.



Métriques d'Évalutation et Outils de Visualisation : AUC

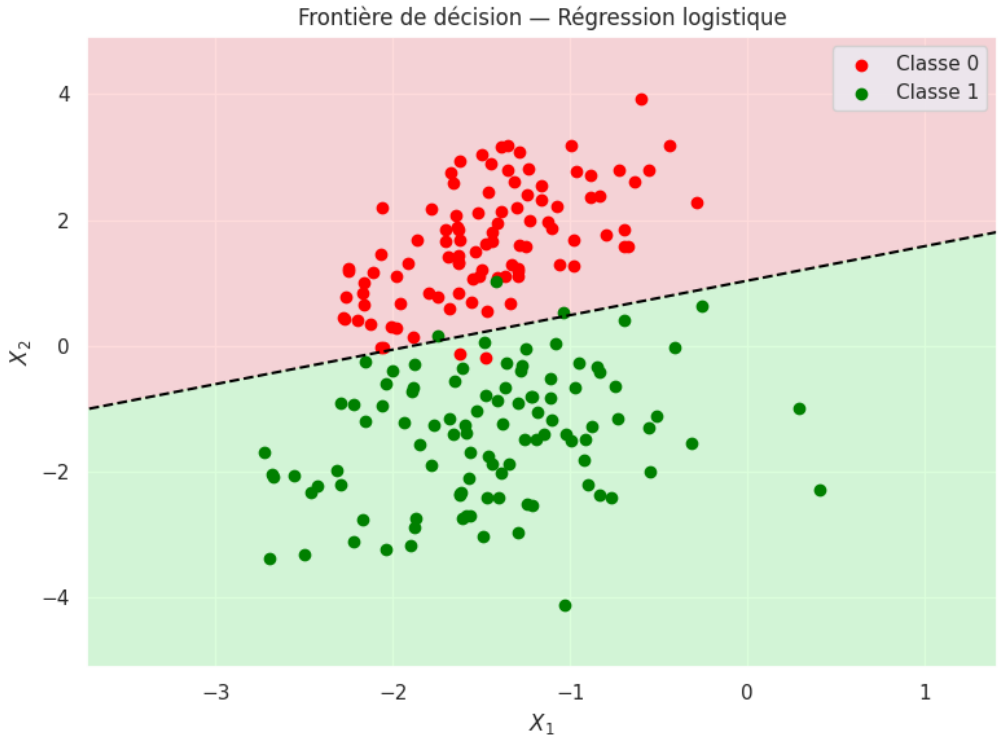
- **Utilité :**
  - **Indépendante du seuil** : on n'a pas à fixer un seuil pour obtenir l'AUC.
  - **Robuste**, même en cas de déséquilibre des classes.
  - Très utilisée pour comparer différents modèles.
- **Limites :**
  - L'AUC ne dit pas tout. Deux modèles peuvent avoir la même AUC mais se comporter différemment selon le seuil.
  - Ne remplace pas les autres métriques (ex. précision, rappel).



Métriques d'Évalutation et Outils de Visualisation : Frontière de Décision (avec PCA)

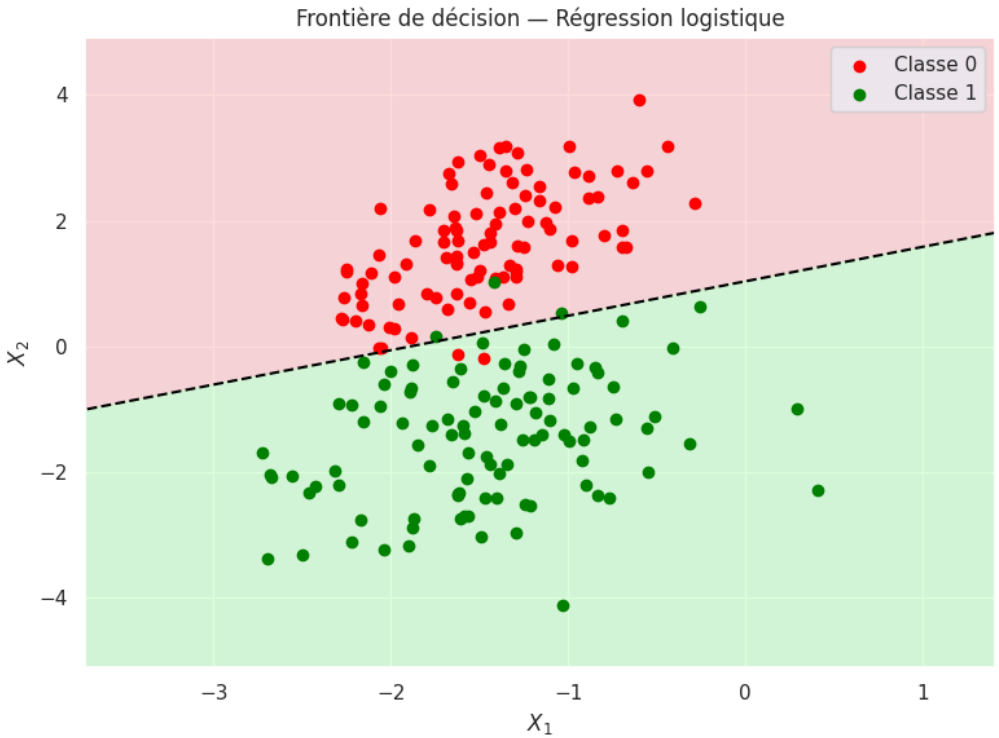
Métriques d'Évalutation et Outils de Visualisation : FD

- **Définition :**
  - La **frontière de décision** est la ligne (ou surface en dimension supérieure) qui **sépare les régions de l'espace des données** où le modèle prédit une classe plutôt qu'une autre.
  - Elle correspond aux points où la **probabilité prédite est égale au seuil**, souvent $p = 0.5$ pour la régression logistique.



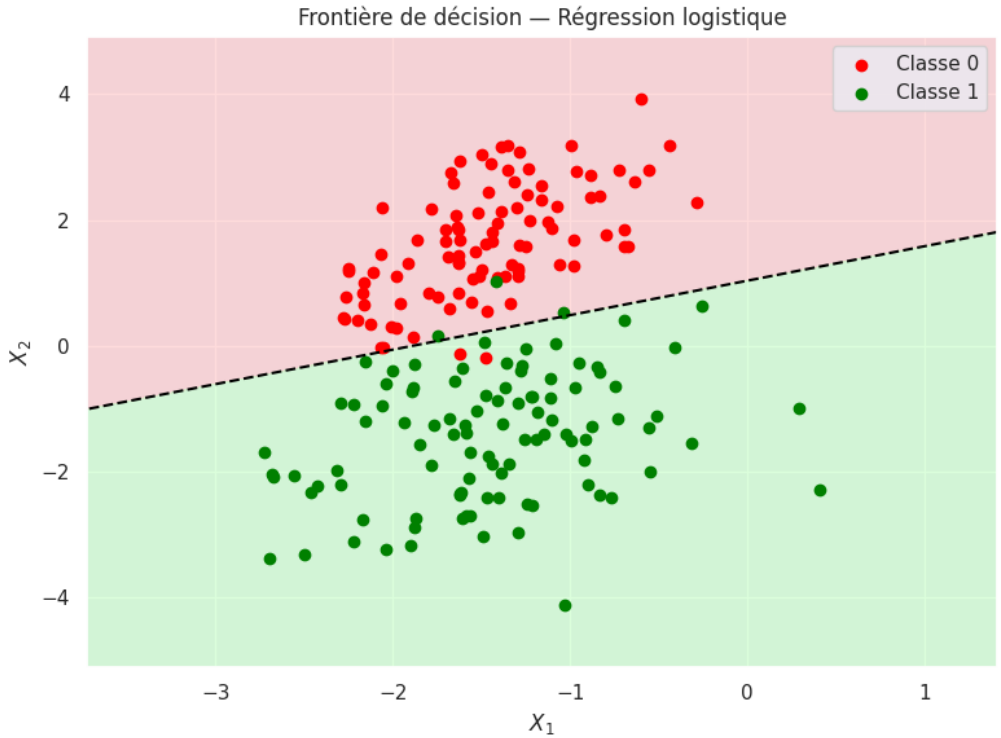
Métriques d'Évalutation et Outils de Visualisation : FD

- **Intuition :**
  - Elle nous aide à comprendre **comment le modèle généralise** à partir des données d'entraînement.
  - Elle montre **quelles zones du plan des caractéristiques** sont associées à chaque classe.



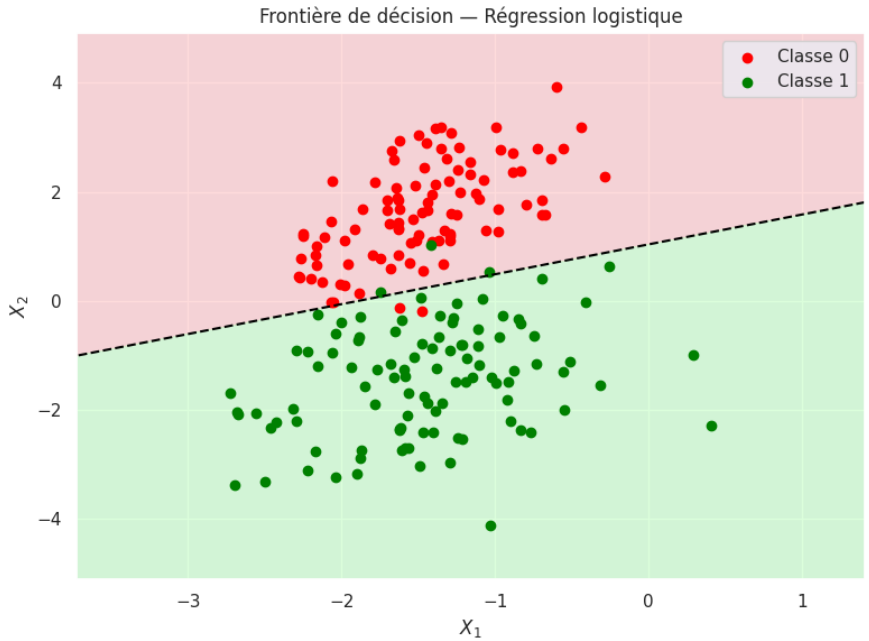
Métriques d'Évalutation et Outils de Visualisation : FD

- **Utilité :**
  - Très utile pour visualiser la décision d'un classifieur sur des jeux de données 2D.
  - Permet de diagnostiquer **le surapprentissage ou la sous-performance**.
  - Donne une idée claire de la **complexité du modèle**.



Métriques d'Évalutation et Outils de Visualisation : PCA

- La visualisation de la frontière de décision est facile quand les données sont en 2D mais dans la plupart des cas on a plus de 2 variables explicatives.
- Dans la plupart des cas on a $\mathbf{X} \in \mathbb{R}^{n \times (p+1)}$ avec $p \gg 2$:
- **Comment visualiser la décision du modèle dans un tel espace?**
- **Réponse :** on projette les données dans un sous-espace de dimension 2 à l'aide de **l'Analyse en Composantes Principales** (PCA : *Principal Component Analysis*).

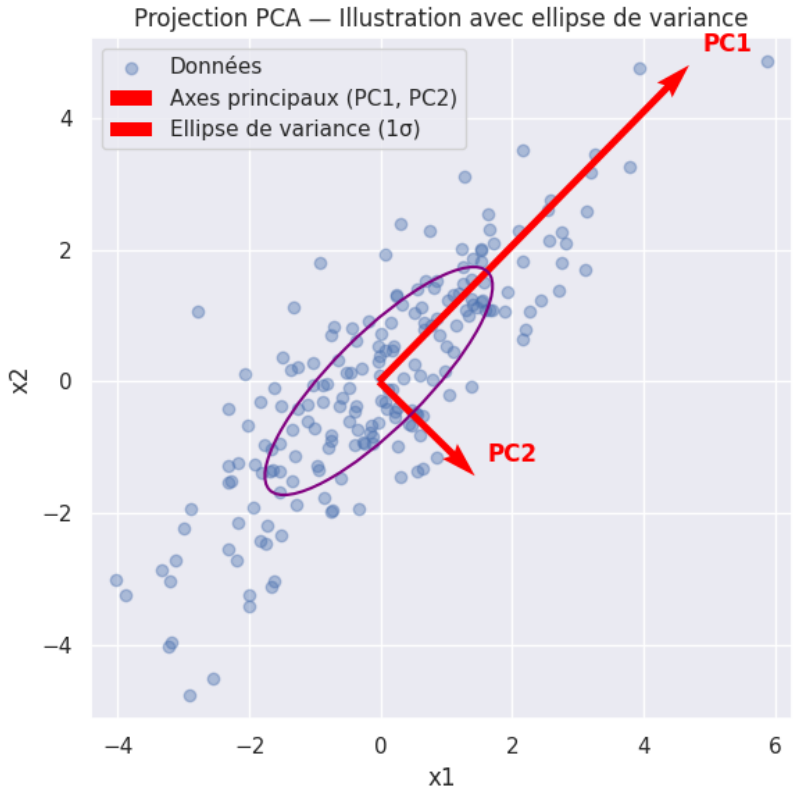


Métriques d'Évalutation et Outils de Visualisation : PCA

- **Analyse en Composantes Principales (PCA) :** Soit une matrice de données centrée $\mathbf{X} \in \mathbb{R}^{n \times p}$, où :
  - n est le nombre d'observations.
  - p est le nombre de variables explicatives.
  - Ici, chaque ligne de $\mathbf{X}$ est un vecteur d'observation centré (moyenne nulle).
- **Objectif :** Trouver une base orthonormée $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p]$ telle que :
  - La projection des données maximise la variance projetée ;
  - Les axes soient orthogonaux entre eux (non corrélés) ;
  - Chaque nouvelle composante soit une combinaison linéaire des variables originales.

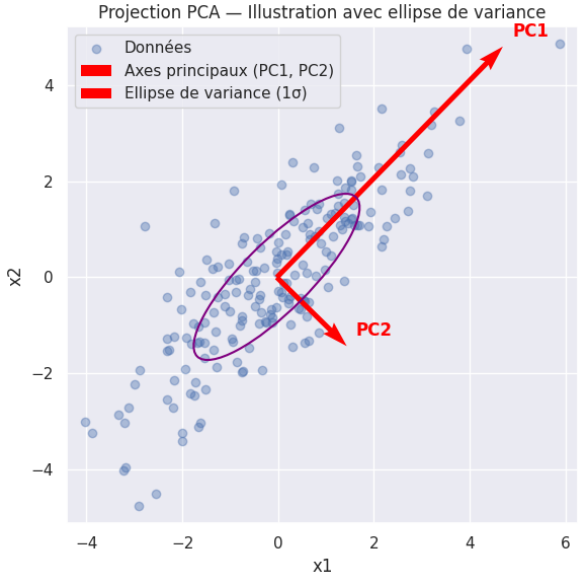
Métriques d'Évalutation et Outils de Visualisation : PCA

- **But de la PCA : Trouver les axes (base de l'espace linéaire) qui capturent le plus d'information possible dans les données.**
- Mais comment mesurer **l'information?**
  - **La variance est utilisée comme proxy de l'information utile.**
  - Plus une projection a de variance, plus elle discrimine les observations, et donc plus elle révèle la structure des données.



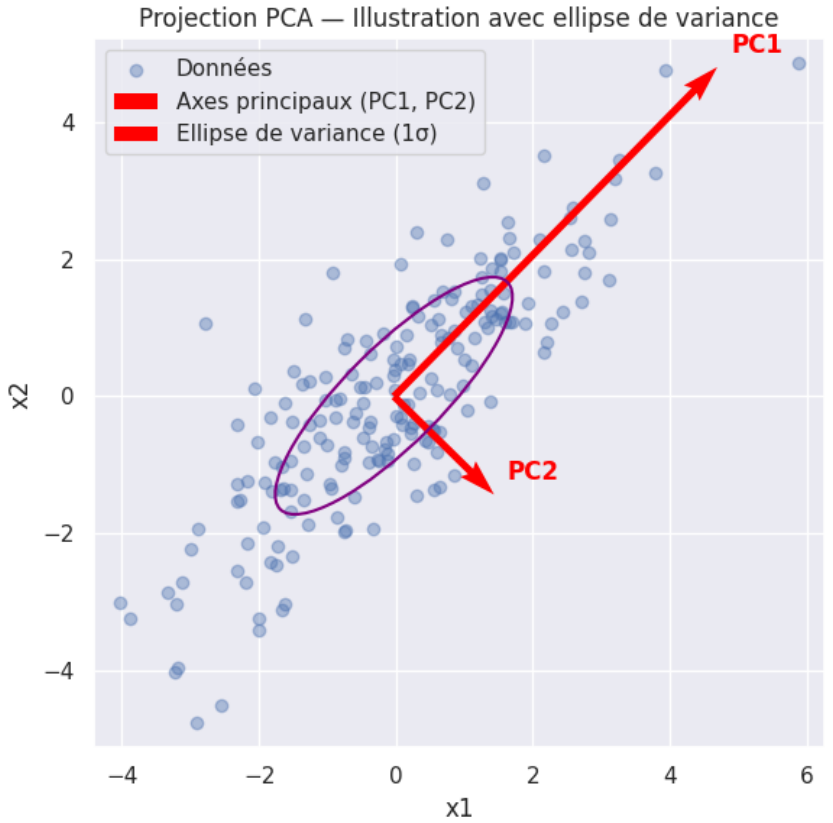
Métriques d'Évalutation et Outils de Visualisation : PCA

- **Objectif 1. Maximiser la Variance = étalement des données**
  - Une faible variance signifie que toutes les données projetées sont proches : peu d'information :
  - Une forte variance signifie que les données projetées sont bien séparées : **plus de nuances à observer.**
  - Elle oriente la première composante dans la direction où la variance est maximale, donc la plus informative.
- **2. Réduire la redondance :**
  - Si deux variables sont très corrélées, elles **répètent la même information.**



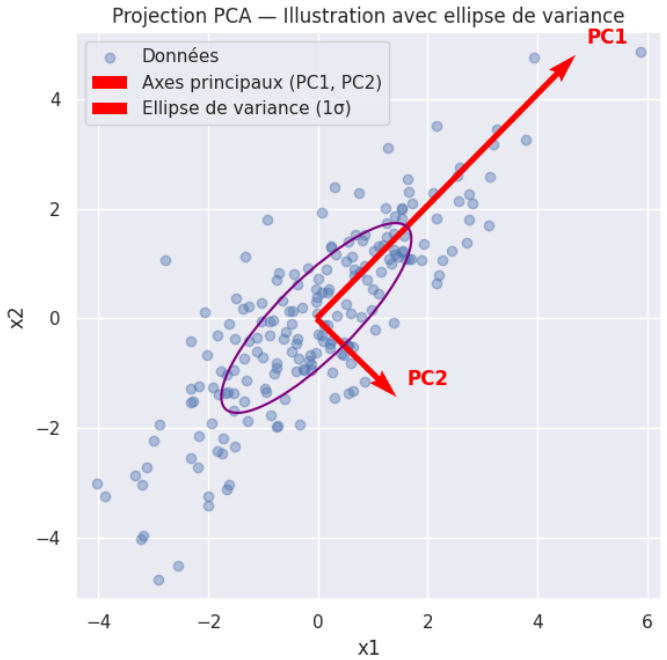
Métriques d'Évalutation et Outils de Visualisation : PCA

- **PCA effectue un changement de repère :**
  - PCA projette les données sur un nouveau système d'axes orthonormés (**linéairement indépendants** et de normes 1).
  - Les variables d'origine sont projetées sur ces axes **tels que leurs variance est maximisée sur chacun de ces axes.**



Métriques d'Évalutation et Outils de Visualisation : PCA

- **PCA effectue un changement de repère :**
- La première composante principale (PC1) est la direction dans laquelle le nuage est le plus étiré.
- En projetant les points sur cette direction, **on conserve la composante la plus importante de l'information.**
- La direction perpendiculaire (PC2) **maximise ensuite la variance restante,** orthogonale à PC1.



Métriques d'Évalutation et Outils de Visualisation : PCA

Objectif : Trouver la direction $\mathbf{u}_1 \in \mathbb{R}^p$ (de norme 1) qui maximise la variance des données projetées.

Formulation mathématique

$$\max_{\mathbf{u}_1 \in \mathbb{R}^p,\, \|\mathbf{u}_1\|=1} \quad \text{Var}(\mathbf{X}\mathbf{u}_1) = \mathbf{u}_1^\top \text{Var}(\mathbf{X})\, \mathbf{u}_1,$$

Métriques d'Évalutation et Outils de Visualisation : PCA

Objectif : Trouver la direction $\mathbf{u}_1 \in \mathbb{R}^p$ (de norme 1) qui maximise la variance des données projetées.

Formulation mathématique

$$\max_{\mathbf{u}_1 \in \mathbb{R}^p,\, \|\mathbf{u}_1\|=1} \text{Var}(\mathbf{X}\mathbf{u}_1) = \mathbf{u}_1^\top \text{Var}(\mathbf{X})\, \mathbf{u}_1,$$

Où $\mathbf{X} \in \mathbb{R}^{n \times p}$ est la matrice de données centrée, et Σ la matrice de covariance empirique tel que $\mathbf{\Sigma} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$.

Métriques d'Évalutation et Outils de Visualisation : PCA

Objectif : Trouver la direction $\mathbf{u}_1 \in \mathbb{R}^p$ (de norme 1) qui maximise la variance des données projetées.

Formulation mathématique

$$\max_{\mathbf{u}_1 \in \mathbb{R}^p,\, \|\mathbf{u}_1\|=1} \text{Var}(\mathbf{X}\mathbf{u}_1) = \mathbf{u}_1^\top \text{Var}(\mathbf{X})\, \mathbf{u}_1,$$

Où $\mathbf{X} \in \mathbb{R}^{n \times p}$ est la matrice de données centrée, et Σ la matrice de covariance empirique tel que $\mathbf{\Sigma} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$.

- La variance des données projetées sur $\mathbf{u}_1$ s'écrit :

$$\text{Var}(\mathbf{X}\mathbf{u}_1) = \mathbf{u}_1^\top \mathbf{\Sigma} \mathbf{u}_1$$

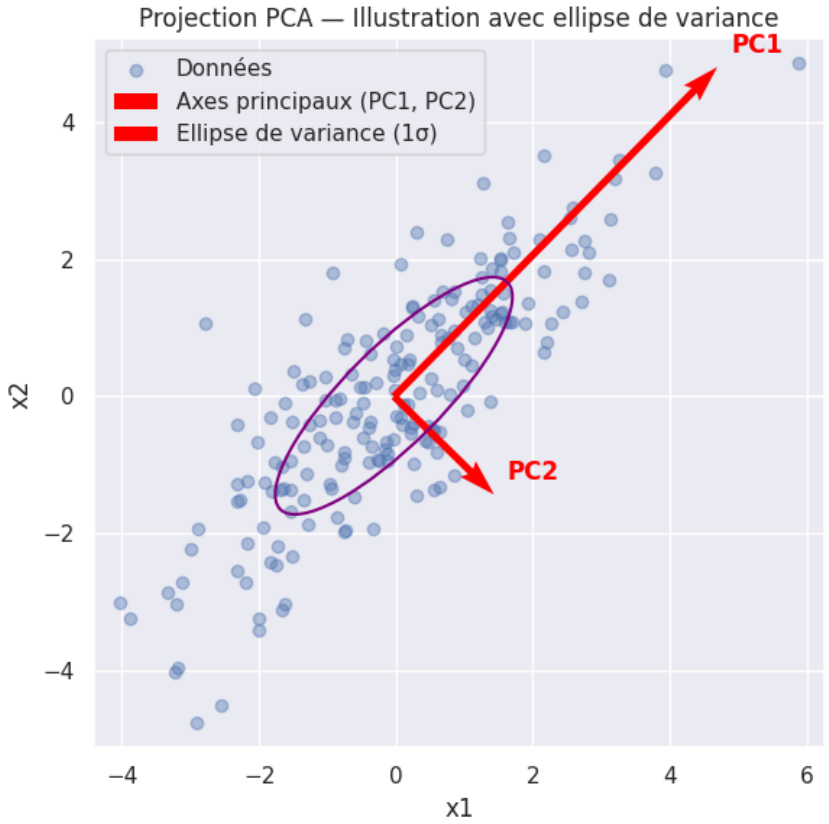
- C'est un problème classique d'optimisation quadratique sous contrainte :

$$\max_{\|\mathbf{u}_1\|=1} \mathbf{u}_1^\top \mathbf{\Sigma} \mathbf{u}_1$$

- **Solution :** $\mathbf{u}_1$ est le **vecteur propre principal de Σ** (associé à la plus grande valeur propre λ_1) **(Preuve ici en Annexe.)**

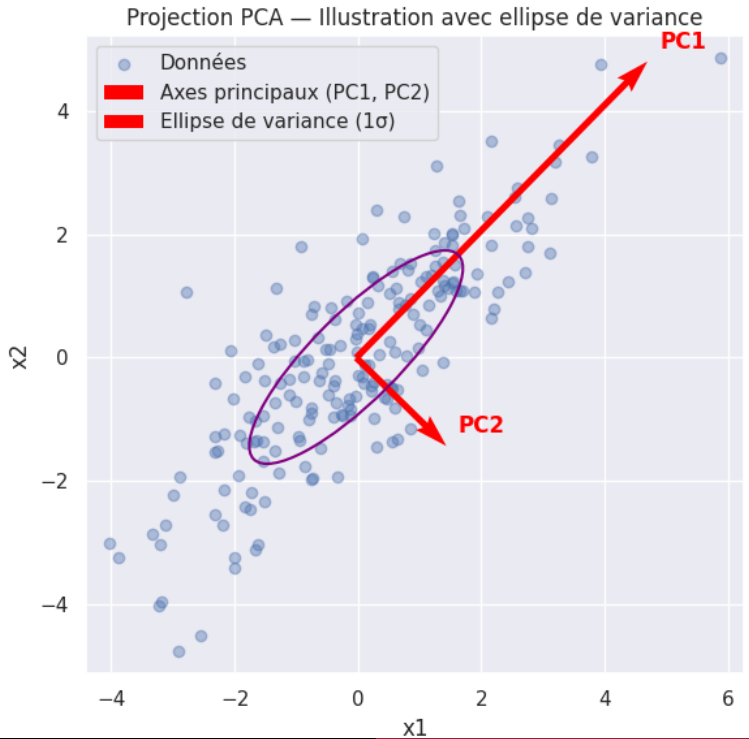
Métriques d'Évalutation et Outils de Visualisation : PCA

- Les données sont projetées sur les directions $\mathbf{u}_1$ et $\mathbf{u}_2$ qui maximisent la dispersion.
- Cela correspond aux axes les plus étirés du nuage de points.
- PCA aligne alors les axes du nouveau repère avec les directions de variation maximale.



Métriques d'Évalutation et Outils de Visualisation : PCA

- Si nous avons beaucoup de variables (par exemple 30 variables explicatives), cela empêche toute visualisation directe ;
- En projetant les données sur les deux premières composantes principales :
  - on capture l'essentiel de la structure du nuage ;
  - on peut visualiser facilement la séparation entre les classes ;
  - on peut dessiner une **frontière de décision** du modèle dans le plan.



Métriques d'Évalutation et Outils de Visualisation : PCA

- **1. Centrer les données** :

$$\mathbf{X}_{\text{centrée}} = \mathbf{X} - \text{mean}(\mathbf{X})$$

- **2. Calculer la matrice de covariance** :

$$\mathbf{\Sigma} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$$

- **3. Décomposer la matrice de covariance** :

$$\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$$

où :

  - $\mathbf{U} \in \mathbb{R}^{p \times p}$: vecteurs propres (axes principaux) ;
  - $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_p)$: valeurs propres (variances expliquées).

- **4. Projeter les données** :

$$\mathbf{Z} = \mathbf{X}\mathbf{U}$$

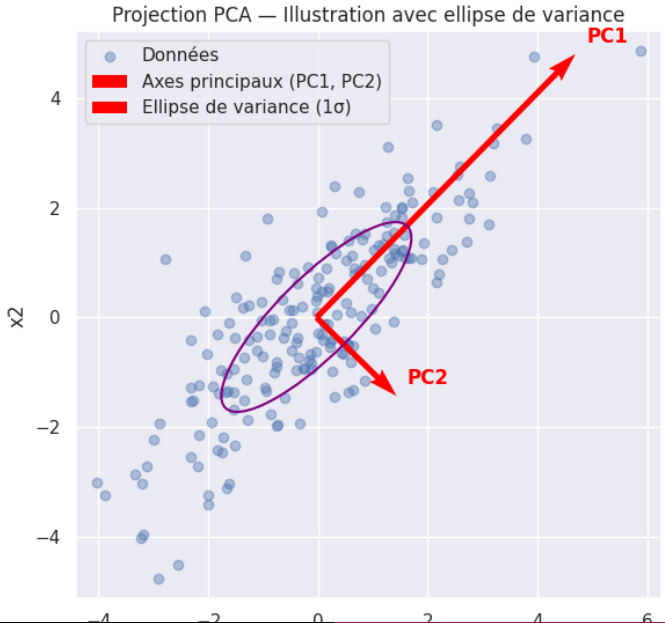
$\mathbf{Z}$ est la représentation des données dans la base des composantes principales.

Métriques d'Évalutation et Outils de Visualisation : PCA

- **Variance expliquée par chaque composante** :

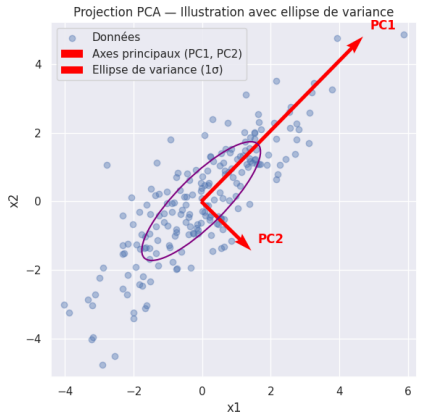
$$\text{Var. expliquée } (\text{PC}_i) = \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j}$$

- **Résumé** :
  - La PCA consiste à résoudre un problème de **maximisation de la variance projetée** sous contrainte d'orthogonalité.
  - Cela revient à effectuer une **décomposition en valeurs propres** de la matrice de covariance.



Métriques d'Évalutation et Outils de Visualisation : PCA

- **PC1 : Première composante principale** :
  - Direction qui maximise la variance totale.
  - Axe "majeur" de variation dans les données.
- **PC2 : Deuxième composante principale** :
  - Direction orthogonale à PC1.
  - Capture la plus grande variance restante.
  - Apporte une vue complémentaire et non redondante.
- **Utilité dans la visualisation** :
  - Ces deux axes permettent de projeter les données dans un **plan 2D informatif**.
  - Facilitent la visualisation des classes et de la frontière de décision.



Exercice: Implémentation des Métriques d'Évalutation et des Outils de Visualisation

Exercice : Métriques d'Évaluation et des Outils de Vis

- **Objectif :** Implémenter manuellement les principales métriques d'évaluation pour la classification binaire ainsi que des fonctions de visualisation pour analyser la performance d'un modèle de régression logistique.
- **Instructions :**
  - Vous allez compléter les fonctions suivantes :
    - `compute_accuracy`, `compute_precision`, `compute_recall`, `compute_f1`, `compute_confusion_matrix`
    - `plot_log_likelihood`, `plot_predictions_vs_observations`, `plot_confusion_matrix`, `plot_roc_curve`
  - Ces fonctions seront appelées dans un pipeline d'entraînement de régression logistique.
  - Vous pouvez utiliser `numpy`, `matplotlib`, `seaborn`, ainsi que les fonctions déjà définies comme `sigmoid`.
  - Le but est de comprendre comment ces métriques et visualisations sont construites "à la main", sans appel aux fonctions de haut niveau de `scikit-learn`.

Exercice : Métriques d'Évaluation et des Outils de Vis

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns; sns.set_theme()
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve, roc_auc_score
# === FONCTIONS UTILITAIRES POUR LA RÉGRESSION LOGISTIQUE ===
def sigmoid(z): # Fonction sigmoïde (fonction d'activation)
    return 1 / (1 + np.exp(-z))
def log_likelihood_logistic(beta, X, y): # Fonction de log-vraisemblance pour la régression logistique
    eta = np.clip(X @ beta, -30, 30)  # stabilisation numérique
    p = sigmoid(eta)
    return np.sum(y * np.log(p + 1e-12) + (1 - y) * np.log(1 - p + 1e-12))
def gradient_logistic(beta, X, y): # Calcul du gradient de la log-vraisemblance
    eta = np.clip(X @ beta, -30, 30)
    p = sigmoid(eta)
    return X.T @ (y - p)
def logistic_regression(X, y, lr=1e-4, max_iter=10000, tol=1e-6, X_val=None, y_val=None): # Descente de gradient
    beta = np.zeros(X.shape[1])
    ll_history_train, ll_history_val = [], []
    for iteration in range(max_iter):
        grad = gradient_logistic(beta, X, y)
        beta += lr * grad
        # Suivi de la log-vraisemblance
        ll_train = log_likelihood_logistic(beta, X, y)
        ll_history_train.append(ll_train)
        if X_val is not None and y_val is not None:
            ll_val = log_likelihood_logistic(beta, X_val, y_val)
            ll_history_val.append(ll_val)
        # Critère d'arrêt : convergence si amélioration faible
        if iteration > 0 and abs(ll_history_train[-1] - ll_history_train[-2]) < tol:
            print(f"Convergence atteinte en {iteration} itérations.")
            break
    return beta, ll_history_train, ll_history_val
```

Exercice : Métriques d'Évaluation et des Outils de Vis

```python
# === MÉTRIQUES D'ÉVALUATION | IMPLÉMENTATION MANUELLE ===
def compute_accuracy(y_true, y_pred):
    return np.mean(_______ == _______)

def compute_precision(y_true, y_pred):
    TP = np.sum((y_true == _______) & (y_pred == _______))
    FP = np.sum(_______)
    return _______ / _______

def compute_recall(y_true, y_pred):
    TP = _______
    FN = _______
    return _______

def compute_f1(precision, recall):
    return 2 * _______

def compute_confusion_matrix(y_true, y_pred):
    TP = _______
    TN = _______
    FP = _______
    FN = _______
    return np.array([[TN, FP], [FN, TP]])

# === VISUALISATIONS ===
def plot_log_likelihood(history_train, history_val, y_train, y_val): # Affiche l'évolution de la log-vraisemblance
    plt.figure(figsize=(8, 5))
    plt.plot(np.array(history_train)/len(y_train), label="Train", color="green")
    plt.plot(np.array(history_val)/len(y_val), label="Validation", color="orange")
    plt.xlabel("Itérations")
    plt.ylabel("Log-vraisemblance normalisée")
    plt.title("Évolution de la log-vraisemblance")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Exercice : Métriques d'Évaluation et des Outils de Vis

```python
# Affiche la relation entre les probabilités prédites et les classes observées
def plot_predictions_vs_observations(p_pred, y_val):
    plt.figure(figsize=(8, 5))
    plt.scatter(p_pred, y_val, alpha=0.6, color='royalblue')
    plt.axvline(x=0.5, color='red', linestyle='--', linewidth=1.5, label='Seuil de décision (0.5)')
    plt.xlabel("Probabilité prédite")
    plt.ylabel("Classe observée")
    plt.title("Régression Logistique | Prédictions vs Observations")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
# Affiche une matrice de confusion
def plot_confusion_matrix(conf_mat):
    plt.figure(figsize=(5, 4))
    sns.heatmap(_______, annot=True, fmt="d", cmap="Blues",
                xticklabels=["Classe 0", "Classe 1"],
                yticklabels=["Classe 0", "Classe 1"])
    plt.xlabel("Prédit")
    plt.ylabel("Réel")
    plt.title("Matrice de confusion")
    plt.tight_layout()
    plt.show()
# Affiche la courbe ROC
def plot_roc_curve(fpr, tpr, auc):
    plt.figure(figsize=(6, 5))
    plt.plot(_______, _______, label=f"AUC = {auc:.3f}", color='darkorange')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel("Faux positifs (FPR)")
    plt.ylabel("Vrais positifs (TPR)")
    plt.title("Courbe ROC")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Exercice : Métriques d'Évaluation et des Outils de Vis

```python
def manual_pca(X, n_components=2):
    # Étape 1 : Centrage des données
    # On soustrait la moyenne de chaque variable (centrage colonne par colonne)
    X_centered = X - _______(_______, axis=0)

    # Étape 2 : Calcul de la matrice de covariance (p x p)
    # On choisit rowvar=False car chaque ligne est une observation
    cov_matrix = np.cov(_______, rowvar=False)

    # Étape 3 : Décomposition spectrale (valeurs propres et vecteurs propres)
    eigvals, eigvecs = np.linalg.eigh(_______)  # méthode stable pour matrices
    ↪  symétriques

    # Étape 4 : Tri des composantes principales par ordre décroissant de variance
    sorted_idx = np.argsort(_______)[::-1]
    eigvals = _______[sorted_idx]
    eigvecs = eigvecs[:, sorted_idx]

    # Étape 5 : Sélection des n premières composantes principales
    components = eigvecs[:, :_______]

    # Étape 6 : Projection des données centrées sur les composantes principales
    X_proj = _______ @ _______
    return X_proj, components
```

Exercice : Métriques d'Évaluation et des Outils de Vis

```python
# Affiche la frontière de décision dans un plan PCA
def plot_decision_boundary_pca(X, y, beta, title="Frontière de décision (PCA manuelle)"):
    # Étape 1 : Réduction en 2D avec PCA manuelle
    X_proj, components = manual_pca(X, n_components=2)
    # Étape 2 : Définition d'une grille couvrant l'espace projeté
    x_min, x_max = X_proj[:, 0].min() - 1, X_proj[:, 0].max() + 1
    y_min, y_max = X_proj[:, 1].min() - 1, X_proj[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300),
                         np.linspace(y_min, y_max, 300))
    grid = np.c_[xx.ravel(), yy.ravel()]  # grille 2D dans l'espace PCA
    # Étape 3 : Inversion de la projection par transposition pour revenir à l'espace original
    # On passe du plan PCA 2D à l'espace des variables initiales
    X_grid_original = grid @ _______.T + np.mean(_______, axis=0)
    # Étape 4 : Ajout de l'intercept pour l'application du modèle linéaire
    X_grid_augmented = np.column_stack((np.ones(X_grid_original.shape[0]), X_grid_original))
    # Étape 5 : Calcul des probabilités prédites par la régression logistique
    def sigmoid(z): return 1 / (1 + np.exp(-z))
    probs = sigmoid(_______ @ _______).reshape(xx.shape)
    # Étape 6 : Tracé graphique de la frontière
    plt.figure(figsize=(7, 5))
    # a) Zones colorées de prédiction
    plt.contourf(xx, yy, probs, levels=[0, 0.5, 1], alpha=0.2, colors=["blue", "orange"])
    # b) Ligne de séparation (probabilité = 0.5)
    plt.contour(xx, yy, probs, levels=[0.5], colors='k', linewidths=1)
    # c) Affichage des données projetées (classes 0 et 1)
    plt.scatter(X_proj[y == 0, 0], X_proj[y == 0, 1], c='blue', label="Classe 0", alpha=0.6)
    plt.scatter(X_proj[y == 1, 0], X_proj[y == 1, 1], c='orange', label="Classe 1", alpha=0.6)
    # d) Mise en forme
    plt.xlabel("Composante principale 1")
    plt.ylabel("Composante principale 2")
    plt.title(title)
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Exercice : Métriques d'Évaluation et des Outils de Vis

```python
# === PIPELINE PRINCIPAL ===
def main():
    # 1. Chargement et préparation des données
    data = load_breast_cancer()
    X_raw, y = data.data, data.target
    feature_names = data.feature_names
    X_train_raw, X_val_raw, y_train, y_val = train_test_split(X_raw, y, test_size=0.2, random_state=8302)
    # 2. Standardisation
    scaler = StandardScaler()
    X_train_std = scaler.fit_transform(X_train_raw)
    X_val_std = scaler.transform(X_val_raw)
    # 3. Ajout de l'intercept
    X_train = np.column_stack((np.ones(X_train_std.shape[0]), X_train_std))
    X_val = np.column_stack((np.ones(X_val_std.shape[0]), X_val_std))
    # 4. Entraînement
    beta_hat, history_train, history_val = logistic_regression(X_train, y_train, X_val=X_val, y_val=y_val)
    # 5. Prédictions
    p_pred = sigmoid(X_val @ beta_hat)
    y_pred_class = (p_pred >= 0.5).astype(int)
    # 6. Affichage des coefficients
    print("Coefficients estimés :")
    for name, coef in zip(['Intercept'] + list(feature_names), beta_hat):
        print(f"  {name} : {coef:.4f}")
    # 7. Visualisation apprentissage et performance
    plot_log_likelihood(history_train, history_val, y_train, y_val)
    plot_predictions_vs_observations(p_pred, y_val)
    # 8. Évaluation manuelle
    accuracy = compute_accuracy(y_val, _______)
    precision = compute_precision(y_val, _______)
    recall = compute_recall(y_val, _______)
    f1 = compute_f1(precision, recall)
    conf_mat = compute_confusion_matrix(y_val, _______)
    fpr, tpr, _ = roc_curve(y_val, p_pred)
    auc = roc_auc_score(y_val, p_pred)
    print("\nMétriques d'évaluation :")
    print(f"  Accuracy  : {accuracy:.3f}")
```

Exercice : Métriques d'Évaluation et des Outils de Vis

```
    print(f"  Précision : {precision:.3f}")
    print(f"  Rappel    : {recall:.3f}")
    print(f"  F1-score  : {f1:.3f}")
    print(f"  AUC       : {auc:.3f}")
    # 9. Visualisations finales
    plot_confusion_matrix(conf_mat)
    plot_roc_curve(fpr, tpr, auc)
    plot_decision_boundary_pca(X_val[:, 1:], y_val, beta_hat)

# Exécution
main()
```

Solution: Implémentation des Métriques d'Évalutation et des Outils de Visualisation

Solution : Métriques d'Évaluation et des Outils de Vis

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns; sns.set_theme()
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_curve, roc_auc_score
# === FONCTIONS UTILITAIRES POUR LA RÉGRESSION LOGISTIQUE ===
def sigmoid(z): # Fonction sigmoïde (fonction d'activation)
    return 1 / (1 + np.exp(-z))
def log_likelihood_logistic(beta, X, y): # Fonction de log-vraisemblance pour la régression logistique
    eta = np.clip(X @ beta, -30, 30)  # stabilisation numérique
    p = sigmoid(eta)
    return np.sum(y * np.log(p + 1e-12) + (1 - y) * np.log(1 - p + 1e-12))
def gradient_logistic(beta, X, y): # Calcul du gradient de la log-vraisemblance
    eta = np.clip(X @ beta, -30, 30)
    p = sigmoid(eta)
    return X.T @ (y - p)
def logistic_regression(X, y, lr=1e-4, max_iter=10000, tol=1e-6, X_val=None, y_val=None): # Descente de gradient
    beta = np.zeros(X.shape[1])
    ll_history_train, ll_history_val = [], []
    for iteration in range(max_iter):
        grad = gradient_logistic(beta, X, y)
        beta += lr * grad
        # Suivi de la log-vraisemblance
        ll_train = log_likelihood_logistic(beta, X, y)
        ll_history_train.append(ll_train)
        if X_val is not None and y_val is not None:
            ll_val = log_likelihood_logistic(beta, X_val, y_val)
            ll_history_val.append(ll_val)
        # Critère d'arrêt : convergence si amélioration faible
        if iteration > 0 and abs(ll_history_train[-1] - ll_history_train[-2]) < tol:
            print(f"Convergence atteinte en {iteration} itérations.")
            break
    return beta, ll_history_train, ll_history_val
```

Solution : Métriques d'Évaluation et des Outils de Vis

```python
# === MÉTRIQUES D'ÉVALUATION | IMPLÉMENTATION MANUELLE ===
def compute_accuracy(y_true, y_pred):
    return np.mean(y_true == y_pred)

def compute_precision(y_true, y_pred):
    TP = np.sum((y_true == 1) & (y_pred == 1))
    FP = np.sum((y_true == 0) & (y_pred == 1))
    return TP / (TP + FP + 1e-12)

def compute_recall(y_true, y_pred):
    TP = np.sum((y_true == 1) & (y_pred == 1))
    FN = np.sum((y_true == 1) & (y_pred == 0))
    return TP / (TP + FN + 1e-12)

def compute_f1(precision, recall):
    return 2 * precision * recall / (precision + recall + 1e-12)

def compute_confusion_matrix(y_true, y_pred):
    TP = np.sum((y_true == 1) & (y_pred == 1))
    TN = np.sum((y_true == 0) & (y_pred == 0))
    FP = np.sum((y_true == 0) & (y_pred == 1))
    FN = np.sum((y_true == 1) & (y_pred == 0))
    return np.array([[TN, FP], [FN, TP]])

# === VISUALISATIONS ===
def plot_log_likelihood(history_train, history_val, y_train, y_val): # Affiche l'évolution de la log-vraisemblance
    plt.figure(figsize=(8, 5))
    plt.plot(np.array(history_train)/len(y_train), label="Train", color="green")
    plt.plot(np.array(history_val)/len(y_val), label="Validation", color="orange")
    plt.xlabel("Itérations")
    plt.ylabel("Log-vraisemblance normalisée")
    plt.title("Évolution de la log-vraisemblance")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Solution : Métriques d'Évaluation et des Outils de Vis

```python
# Affiche la relation entre les probabilités prédites et les classes observées
def plot_predictions_vs_observations(p_pred, y_val):
    plt.figure(figsize=(8, 5))
    plt.scatter(p_pred, y_val, alpha=0.6, color='royalblue')
    plt.axvline(x=0.5, color='red', linestyle='--', linewidth=1.5, label='Seuil de décision (0.5)')
    plt.xlabel("Probabilité prédite")
    plt.ylabel("Classe observée")
    plt.title("Régression Logistique | Prédictions vs Observations")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
# Affiche une matrice de confusion
def plot_confusion_matrix(conf_mat):
    plt.figure(figsize=(5, 4))
    sns.heatmap(conf_mat, annot=True, fmt="d", cmap="Blues",
                xticklabels=["Classe 0", "Classe 1"],
                yticklabels=["Classe 0", "Classe 1"])
    plt.xlabel("Prédit")
    plt.ylabel("Réel")
    plt.title("Matrice de confusion")
    plt.tight_layout()
    plt.show()
# Affiche la courbe ROC
def plot_roc_curve(fpr, tpr, auc):
    plt.figure(figsize=(6, 5))
    plt.plot(fpr, tpr, label=f"AUC = {auc:.3f}", color='darkorange')
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlabel("Faux positifs (FPR)")
    plt.ylabel("Vrais positifs (TPR)")
    plt.title("Courbe ROC")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Solution : Métriques d'Évaluation et des Outils de Vis

```python
def manual_pca(X, n_components=2):
    # Étape 1 : Centrage des données
    # On soustrait la moyenne de chaque variable (centrage colonne par colonne)
    X_centered = X - np.mean(X, axis=0)

    # Étape 2 : Calcul de la matrice de covariance (p x p)
    # On choisit rowvar=False car chaque ligne est une observation
    cov_matrix = np.cov(X_centered, rowvar=False)

    # Étape 3 : Décomposition spectrale (valeurs propres et vecteurs propres)
    eigvals, eigvecs = np.linalg.eigh(cov_matrix)  # méthode stable pour matrices
    ↪  symétriques

    # Étape 4 : Tri des composantes principales par ordre décroissant de variance
    sorted_idx = np.argsort(eigvals)[::-1]
    eigvals = eigvals[sorted_idx]
    eigvecs = eigvecs[:, sorted_idx]

    # Étape 5 : Sélection des n premières composantes principales
    components = eigvecs[:, :n_components]

    # Étape 6 : Projection des données centrées sur les composantes principales
    X_proj = X_centered @ components
    return X_proj, components
```

Solution : Métriques d'Évaluation et des Outils de Vis

```python
# Affiche la frontière de décision dans un plan PCA
def plot_decision_boundary_pca(X, y, beta, title="Frontière de décision (PCA manuelle)"):
    # Étape 1 : Réduction en 2D avec PCA manuelle
    X_proj, components = manual_pca(X, n_components=2)
    # Étape 2 : Définition d'une grille couvrant l'espace projeté
    x_min, x_max = X_proj[:, 0].min() - 1, X_proj[:, 0].max() + 1
    y_min, y_max = X_proj[:, 1].min() - 1, X_proj[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300),
                         np.linspace(y_min, y_max, 300))
    grid = np.c_[xx.ravel(), yy.ravel()]  # grille 2D dans l'espace PCA
    # Étape 3 : Inversion de la projection par transposition pour revenir à l'espace original
    # On passe du plan PCA 2D à l'espace des variables initiales
    X_grid_original = grid @ components.T + np.mean(X, axis=0)
    # Étape 4 : Ajout de l'intercept pour l'application du modèle linéaire
    X_grid_augmented = np.column_stack((np.ones(X_grid_original.shape[0]), X_grid_original))
    # Étape 5 : Calcul des probabilités prédites par la régression logistique
    def sigmoid(z): return 1 / (1 + np.exp(-z))
    probs = sigmoid(X_grid_augmented @ beta).reshape(xx.shape)
    # Étape 6 : Tracé graphique de la frontière
    plt.figure(figsize=(7, 5))
    # a) Zones colorées de prédiction
    plt.contourf(xx, yy, probs, levels=[0, 0.5, 1], alpha=0.2, colors=["blue", "orange"])
    # b) Ligne de séparation (probabilité = 0.5)
    plt.contour(xx, yy, probs, levels=[0.5], colors='k', linewidths=1)
    # c) Affichage des données projetées (classes 0 et 1)
    plt.scatter(X_proj[y == 0, 0], X_proj[y == 0, 1], c='blue', label="Classe 0", alpha=0.6)
    plt.scatter(X_proj[y == 1, 0], X_proj[y == 1, 1], c='orange', label="Classe 1", alpha=0.6)
    # d) Mise en forme
    plt.xlabel("Composante principale 1")
    plt.ylabel("Composante principale 2")
    plt.title(title)
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Solution : Métriques d'Évaluation et des Outils de Vis

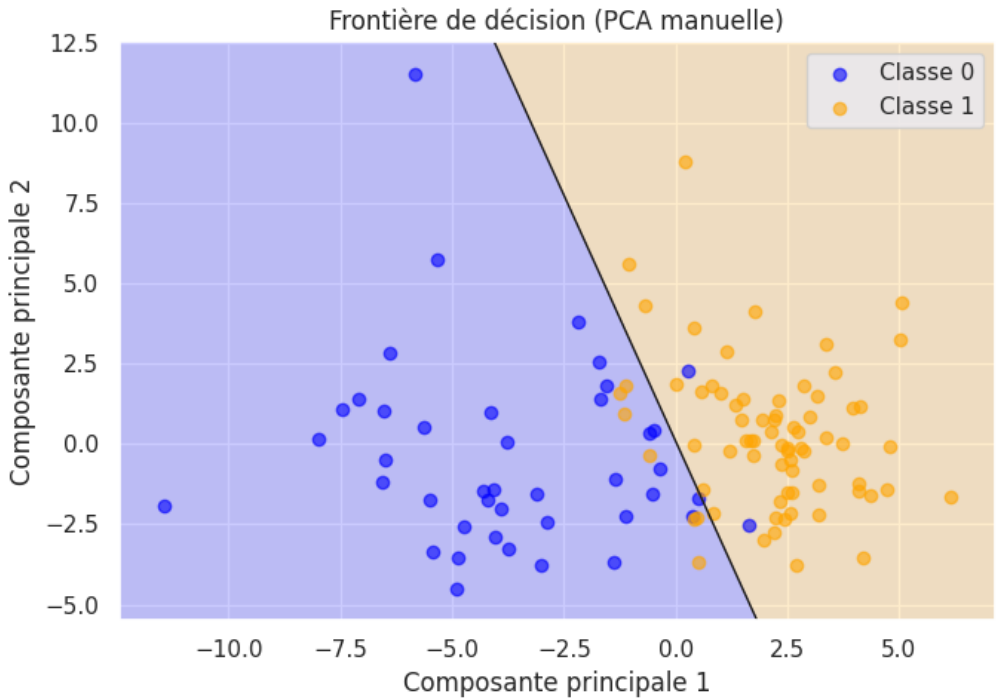
```python
# === PIPELINE PRINCIPAL ===
def main():
    # 1. Chargement et préparation des données
    data = load_breast_cancer()
    X_raw, y = data.data, data.target
    feature_names = data.feature_names
    X_train_raw, X_val_raw, y_train, y_val = train_test_split(X_raw, y, test_size=0.2, random_state=8302)
    # 2. Standardisation
    scaler = StandardScaler()
    X_train_std = scaler.fit_transform(X_train_raw)
    X_val_std = scaler.transform(X_val_raw)
    # 3. Ajout de l'intercept
    X_train = np.column_stack((np.ones(X_train_std.shape[0]), X_train_std))
    X_val = np.column_stack((np.ones(X_val_std.shape[0]), X_val_std))
    # 4. Entraînement
    beta_hat, history_train, history_val = logistic_regression(X_train, y_train, X_val=X_val, y_val=y_val)
    # 5. Prédictions
    p_pred = sigmoid(X_val @ beta_hat)
    y_pred_class = (p_pred >= 0.5).astype(int)
    # 6. Affichage des coefficients
    print("Coefficients estimés :")
    for name, coef in zip(['Intercept'] + list(feature_names), beta_hat):
        print(f"  {name} : {coef:.4f}")
    # 7. Visualisation apprentissage et performance
    plot_log_likelihood(history_train, history_val, y_train, y_val)
    plot_predictions_vs_observations(p_pred, y_val)
    # 8. Évaluation manuelle
    accuracy = compute_accuracy(y_val, y_pred_class)
    precision = compute_precision(y_val, y_pred_class)
    recall = compute_recall(y_val, y_pred_class)
    f1 = compute_f1(precision, recall)
    conf_mat = compute_confusion_matrix(y_val, y_pred_class)
    fpr, tpr, _ = roc_curve(y_val, p_pred)
    auc = roc_auc_score(y_val, p_pred)
    print("\nMétriques d'évaluation :")
    print(f"  Accuracy  : {accuracy:.3f}")
```

Solution : Métriques d'Évaluation et des Outils de Vis

```
    print(f"  Précision : {precision:.3f}")
    print(f"  Rappel    : {recall:.3f}")
    print(f"  F1-score  : {f1:.3f}")
    print(f"  AUC       : {auc:.3f}")
    # 9. Visualisations finales
    plot_confusion_matrix(conf_mat)
    plot_roc_curve(fpr, tpr, auc)
    plot_decision_boundary_pca(X_val[:, 1:], y_val, beta_hat)

# Exécution
main()
```

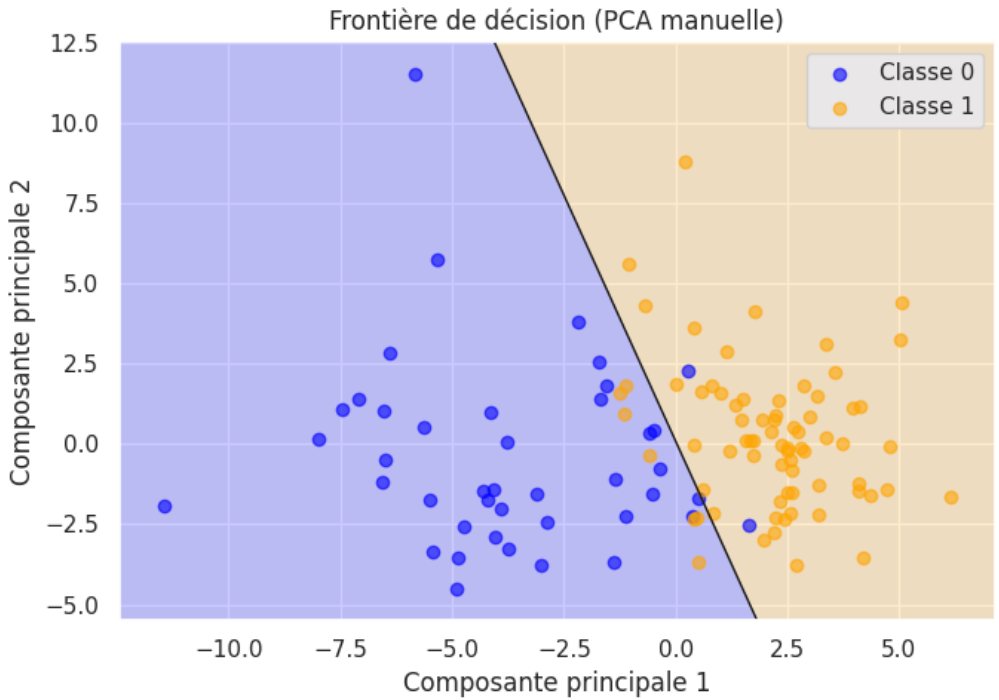
Solution : Métriques d'Évaluation et des Outils de Vis

- **Ce que l'on observe :**
  - Les classes sont bien séparées dans le plan des deux premières composantes principales.
  - La ligne noire correspond à la frontière de décision du modèle appris.
  - La zone bleue correspond aux prédictions pour la classe 0, et l'orange à la classe 1.



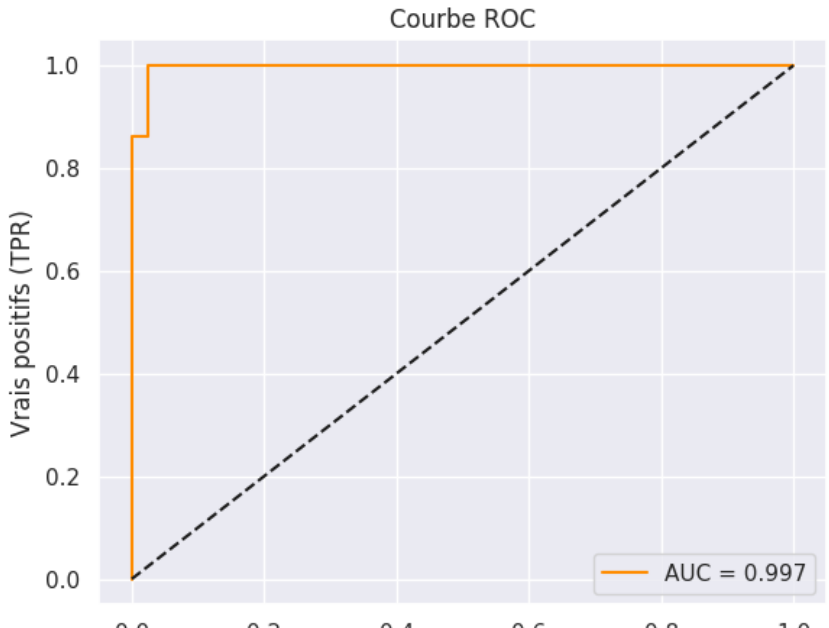
Solution : Métriques d'Évaluation et des Outils de Vis

- **Interprétation :**
  - La projection PCA est très informative ici : elle permet de visualiser que les données sont quasi-linéairement séparables.
  - La frontière est cohérente avec les données projetées : très peu de points sont mal classés visuellement.



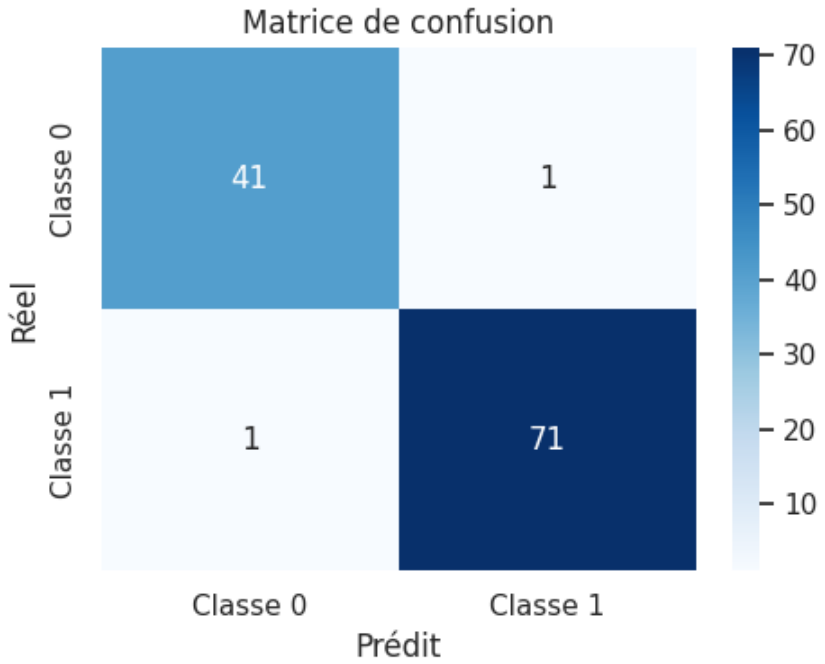
Solution : Métriques d'Évaluation et des Outils de Vis

- **Ce que l'on observe :**
  - La courbe ROC est presque collée au coin supérieur gauche, ce qui est excellent.
  - L'AUC = 0.997, très proche de 1.
- **Interprétation :**
  - Le modèle distingue très bien les deux classes (quasi-parfaite séparation).
  - Cela confirme ce qu'on a vu avec la frontière de décision : le classifieur est très performant sur ces données.



Solution : Métriques d'Évaluation et des Outils de Vis

- **Matrice observée :**
  - Réel : 0 $\rightarrow$ Prédit : 0 = 41, Prédit : 1 = 1
  - Réel : 1 $\rightarrow$ Prédit : 0 = 1, Prédit : 1 = 71
- **Ce que l'on observe :**
  - Seulement 2 erreurs de classification (1 faux positif + 1 faux négatif).
  - 41 + 71 = 112 bonnes prédictions sur 114 exemples.



Solution : Métriques d'Évaluation et des Outils de Vis

- **Métriques :**
  - Accuracy : $(41 + 71)/114 \approx 0.982$
  - Précision (classe 1) : $71/(71 + 1) \approx 0.986$
  - Rappel (classe 1) : $71/(71 + 1) \approx 0.986$
  - F1-score : ≈ 0.986
- **Interprétation :**
  - Peu d'erreurs et ces erreurs sont symétriques (1 FP, 1 FN).
  - La précision et le rappel excellents $\Rightarrow$ F1-score élevé.

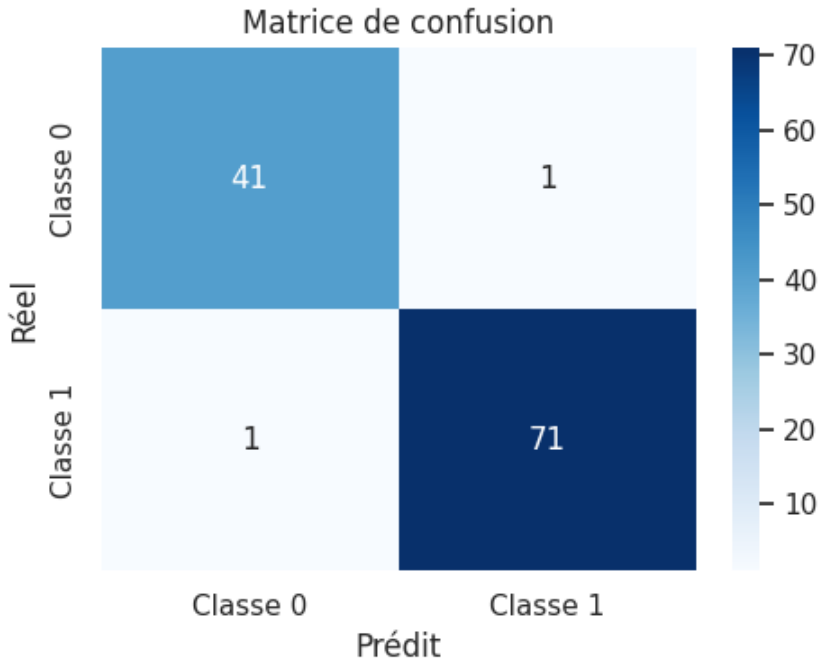


Table des Matières

1 Modèles Linéaires Généralisés

2 Regression Logistique

3 Analyse Linéaire Discriminante

4 Annexe

Analyse Linéaire Discriminante (LDA) et Machines à Vecteurs de Support (SVMs)

Linear Discriminant Analysis (LDA) & Support Vector Machines (SVMs)

Analyse Linéaire Discriminante : Introduction

- L'analyse linéaire discriminante (LDA) est une méthode statistique de classification supervisée.
- Elle permet de classer des objets, personnes ou événements en groupes prédéfinis à partir de variables explicatives.
- L'analyse linéaire discriminante cherche à trouver des directions de projection dans l'espace des variables qui **maximisent la variance entre les classes (inter-classes)** et **minimisent la variance au sein des classes (intra-classe)**.
- Ces directions sont appelées les **composantes discriminantes**, et permettent de mieux séparer les groupes dans un espace de dimension réduite.
- Les données sont ainsi projetées sur ces axes discriminants pour faciliter la classification.

Analyse Linéaire Discriminante : Introduction

- L'analyse linéaire discriminante est largement utilisée dans plusieurs domaines :
  - Reconnaissance de formes et d'images ;
  - Biostatistique (diagnostic médical, classification de profils) ;
  - Finance (analyse de risque et classification de crédit) ;
  - Apprentissage automatique, comme technique de réduction de dimension supervisée.
- Comme la PCA, elle permet de réduire la dimension des données tout en conservant l'information pertinente.
- **Différence clé entre PCA et LDA:** la PCA est non supervisée et ne prend pas en compte les étiquettes de classe, alors que la LDA utilise les labels pour **maximiser la séparabilité entre les classes.**
- La LDA identifie donc des directions pertinentes **pour la classification**, contrairement à la PCA qui vise seulement à expliquer la variance globale.

Analyse Linéaire Discriminante : Introduction

- L'analyse discriminante repose sur certaines hypothèses fondamentales :
  - Les données suivent une **distribution normale multivariée** dans chaque classe.
  - Les différentes classes partagent une **même matrice de covariance** (homoscédasticité).
- Elle offre une méthode à la fois **statistiquement rigoureuse** et **interprétable**, prenant en compte la variabilité intra- et inter-classes.
- C'est une approche pertinente pour des problèmes où la séparation linéaire entre classes est raisonnablement justifiée.
- Ces hypothèses permettent d'utiliser la règle de Bayes pour établir une frontière de décision **linéaire** entre les classes.

LDA avec Formulation Bayesienne

LDA : Modèles Discriminatif vs Génératif

- La **régression logistique** est un modèle discriminatif :
  - Elle modélise directement la probabilité conditionnelle $P(Y_i = y \mid \mathbf{X}_i = \mathbf{x})$.
  - L'objectif est de séparer les classes en ajustant un modèle qui maximise la vraisemblance.
- **L'analyse linéaire discriminante (LDA)** est un modèle génératif :
  - Elle commence par modéliser la probabilité des données dans chaque classe, c'est-à-dire $P(\mathbf{X}_i = \mathbf{x} \mid Y_i = y)$, ainsi que la probabilité a priori $P(Y_i = y)$;
  - Puis elle applique la règle de Bayes pour en déduire la probabilité a posteriori $P(Y_i = y \mid \mathbf{X}_i = \mathbf{x})$.
- **Approche Probabiliste :** LDA exploite une hypothèse probabiliste sur la distribution des données pour effectuer la classification.

LDA : Hypothèses du Modèle

- On suppose que les données $\mathbf{X}_i$ suivent une loi normale multivariée conditionnellement à la classe Y_i :

$$\mathbf{X}_i \mid Y_i = y \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma})$$

- Chaque classe y possède sa propre moyenne $\boldsymbol{\mu}_y$, mais toutes les classes partagent la même matrice de covariance $\boldsymbol{\Sigma}$ (hypothèse d'homoscédasticité).
- Les probabilités a priori des classes sont données par :

$$P(Y_i = y) = p_y \quad \text{avec} \sum_y p_y = 1$$

- Ces hypothèses permettent une classification efficace à l'aide de la règle de Bayes.

LDA avec Formulation Bayesienne

- Sous les hypothèses de LDA, la densité des données $\mathbf{X}_i$ sachant leur classe $Y_i = y$ est donnée par la loi normale multivariée :

$$P(\mathbf{X}_i = \mathbf{x} \mid Y_i = y) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_y)\right).$$

- Cette expression combine :
  - Une **partie constante** liée à la dimension et à la covariance.
  - Une **partie exponentielle** qui mesure la distance entre $\mathbf{x}$ et la moyenne de la classe y, pondérée par la matrice de covariance.
- Pour prédire la classe d'une observation $\mathbf{x}$, on utilise la règle de Bayes :

$$P(Y_i = y \mid \mathbf{X}_i = \mathbf{x}) \propto P(Y_i = y) \cdot P(\mathbf{X}_i = \mathbf{x} \mid Y_i = y)$$

LDA avec Formulation Bayesienne

- En prenant le logarithme de cette expression, on obtient :

$$\log\left[P(Y_i = y \mid \mathbf{X}_i = \mathbf{x})\right] = \log(p_y) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_y) + C$$

- **Remarque :** La constante C est commune à toutes les classes et correspond à :

$$C = -\frac{p}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}|$$

- Elle n'intervient pas dans la décision finale (comparaison entre classes), car elle est la même pour toutes les classes.
- Le log-score $\log\left[P(Y_i = y \mid \mathbf{X}_i = \mathbf{x})\right]$ nous permet de construire une fonction discriminante linéaire pour chaque classe.

LDA avec Formulation Bayesienne

- Le log-score $\log\left[P(Y_i = y \mid \mathbf{X}_i = \mathbf{x})\right]$ est donné par :

$$\log\left[P(Y_i = y \mid \mathbf{X}_i = \mathbf{x})\right] = \log(p_y) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_y) + C$$

- La fonction discriminante associée à une classe y est définie par :

$$\delta_y(\mathbf{x}) = \cancel{-\frac{1}{2}\underbrace{\mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\mathbf{x}}_{\text{ne dépend pas de } y}} + \mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_y - \frac{1}{2}\boldsymbol{\mu}_y^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_y + \log(p_y)$$

$$= \mathbf{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_y - \frac{1}{2}\boldsymbol{\mu}_y^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_y + \log(p_y).$$

- Cette fonction est **affine en** $\mathbf{x}$: elle dépend linéairement de l'observation $\mathbf{x}$.
- La prédiction se fait en choisissant la classe qui maximise ce score :

$$\widehat{Y}_i = \arg\max_y \ \delta_y(\mathbf{X}_i)$$

- Les frontières de décision sont donc des **hyperplans linéaires** qui séparent les classes dans l'espace des variables.

LDA avec Formulation Bayesienne

- Dans le cas binaire $Y \in \{0, 1\}$, on prédit la classe $Y = 1$ si :

$$\delta_1(\mathbf{x}) > \delta_0(\mathbf{x}) \quad \Leftrightarrow \quad \widehat{Y} = 1$$

- La frontière de décision est donc définie par l'ensemble :

$$\{\mathbf{x} \in \mathbb{R}^p \mid \delta_1(\mathbf{x}) = \delta_0(\mathbf{x})\}$$

- On prédit $Y = 1$ si : $\delta_1(\mathbf{x}) - \delta_0(\mathbf{x}) > 0$
- Or :

$$\delta_1(\mathbf{x}) - \delta_0(\mathbf{x}) = \mathbf{x}^\top \underbrace{\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}_{\mathbf{w} \in \mathbb{R}^p} \underbrace{-\frac{1}{2}\left(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_0\right) + \log\left(\frac{p_1}{p_0}\right)}_{-c \in \mathbb{R}}.$$

LDA avec Formulation Bayesienne

- On prédit $Y = 1$ si : $\delta_1(\mathbf{x}) - \delta_0(\mathbf{x}) > 0$
- Or :

$$\delta_1(\mathbf{x}) - \delta_0(\mathbf{x}) = \mathbf{x}^\top \underbrace{\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}_{\mathbf{w} \in \mathbb{R}^p} \underbrace{-\frac{1}{2}\left(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0\right) + \log\left(\frac{p_1}{p_0}\right)}_{-c \in \mathbb{R}}.$$

- On note :

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \in \mathbb{R}^p.$$

$$c = \frac{1}{2}\left(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0\right) - \log\left(\frac{p_1}{p_0}\right) \in \mathbb{R}.$$

- La frontière de décision est donc linéaire :

$$\widehat{Y} = 1 \quad \text{si et seulement si} \quad \delta_1(\mathbf{x}) - \delta_0(\mathbf{x}) > 0 \Leftrightarrow \mathbf{x}^\top \mathbf{w} > c.$$

LDA avec Formulation Bayesienne

- **Fonction Discriminante** :

$$\delta_y(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_y - \frac{1}{2} \boldsymbol{\mu}_y^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_y + \log(p_y)$$

- **Équation de la Frontière de Décision** :

$$\mathbf{x}^\top \underbrace{\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}_{\mathbf{w} \in \mathbb{R}^p} = c \quad \text{(constante)}$$

- **Conclusion** : La règle de Bayes donne une séparation par un hyperplan :

$$\widehat{Y} = \arg\max_y \ \delta_y(\mathbf{x})$$

LDA avec Formulation Bayesienne

- L'expression mathématique de la frontière de décision dans le cadre de l'analyse discriminante linéaire (LDA) Bayésienne pour 2 classes $Y \in \{0, 1\}$ est alors une équation linéaire de la forme :
- **Équation de la Frontière de Décision** :

$$\mathbf{x}^\top \underbrace{\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}_{\mathbf{w} \in \mathbb{R}^p} = c \quad \text{(constante)}.$$

$$\Rightarrow \mathbf{x}^\top \mathbf{w} = c \quad \text{(constante)}.$$

- Où :
  - $\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$ est un vecteur normal à l'hyperplan de la frontière de décision $H = \left\{\mathbf{x} \in \mathbb{R}^p \mid \mathbf{w}^\top \mathbf{x} = c\right\}$.
  - $c = \frac{1}{2}\left(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_0\right) - \log\left(\frac{p_1}{p_0}\right)$
- Cette équation définit un **hyperplan** (et en 2D, une droite) dans l'espace des variables explicatives.
- Il s'agit de l'endroit où les fonctions discriminantes pour chaque classe sont égales :

$$\delta_1(\mathbf{x}) = \delta_0(\mathbf{x}).$$

LDA avec Formulation Bayesienne

- Si $\mathbf{x} = (x_1, x_2)^\top$ et $\mathbf{w} = (w_1, w_2)^\top$, alors l'équation de la frontière devient :

$$x_1 w_1 + x_2 w_2 = c \quad \Rightarrow \quad x_2 = \frac{c - w_1 x_1}{w_2}$$

- Cela permet de visualiser la frontière comme une droite dans le plan (x_1, x_2), qui sépare les régions associées aux deux classes.
- La normalité de $\mathbf{w}$ à l'hyperplan $H = \left\{\mathbf{x} \in \mathbb{R}^p \;\middle|\; \mathbf{w}^\top \mathbf{x} = c\right\}$ qui définit la frontière de décision permet d'interpréter la direction de séparation.
- Cela signifie :
  - $\mathbf{w}$ est perpendiculaire à toute variation $\mathbf{v}$ à l'intérieur de H, i.e. $\mathbf{v}^\top \mathbf{w} = 0$
  - $\mathbf{w}$ indique la direction dans laquelle la valeur de $\mathbf{x}^\top \mathbf{w}$ augmente la probabilité d'avoir $\hat{Y} = 1$.

LDA avec Formulation Bayesienne

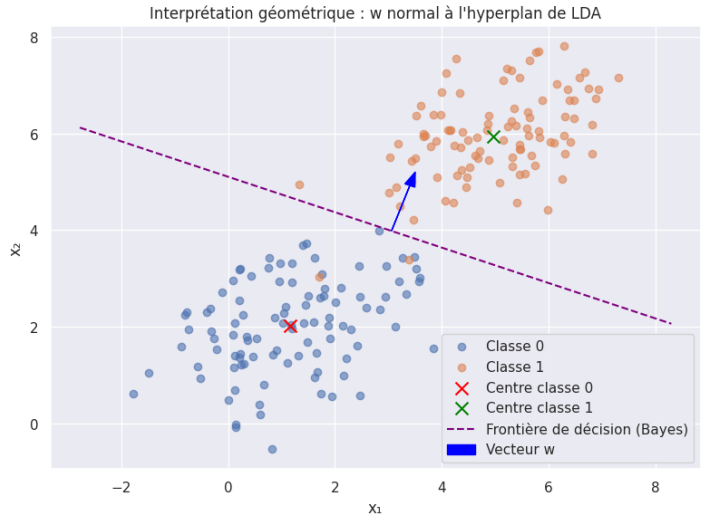
- Pour mieux comprendre, soit 2 points $\mathbf{x}_1, \mathbf{x}_2 \in H = \left\{\mathbf{x} \in \mathbb{R}^p \;\middle|\; \mathbf{w}^\top \mathbf{x} = c\right\}$ tels que :

$$\mathbf{x}_1^\top \mathbf{w} = c \quad \text{et} \quad \mathbf{x}_2^\top \mathbf{w} = c$$

- Alors, $\mathbf{v} = \mathbf{x}_1 - \mathbf{x}_2$ **satisfait :**

$$\mathbf{v}^\top \mathbf{w} = (\mathbf{x}_1 - \mathbf{x}_2)^\top \mathbf{w} = \mathbf{x}_1{}^\top \mathbf{w} - \mathbf{x}_2{}^\top \mathbf{w} = c - c = 0$$

- Donc, $\mathbf{v} \perp \mathbf{w} \quad \forall \quad \mathbf{v} \in H = \left\{\mathbf{x} \in \mathbb{R}^p \;\middle|\; \mathbf{w}^\top \mathbf{x} = c\right\}$. Ceci veut dire que $\mathbf{w}$ est orthogonal à tout vecteur contenu dans l'hyperplan de la frontière de décision H. On dit que $\mathbf{w}$ est normal à l'hyperplan H.

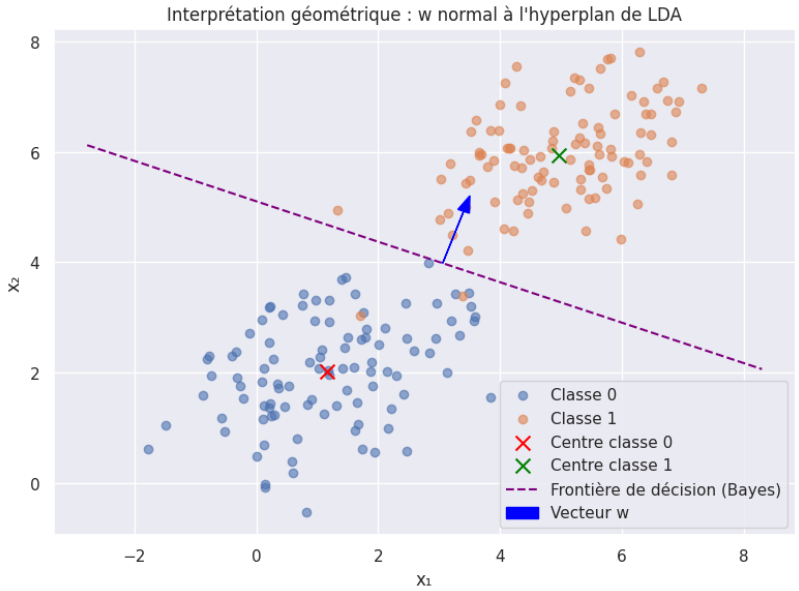


LDA avec Formulation Bayesienne

- La frontière de décision est donnée par : $\mathbf{x}^\top \underbrace{\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}_{\mathbf{w} \in \mathbb{R}^p} = c.$
- On prédit :

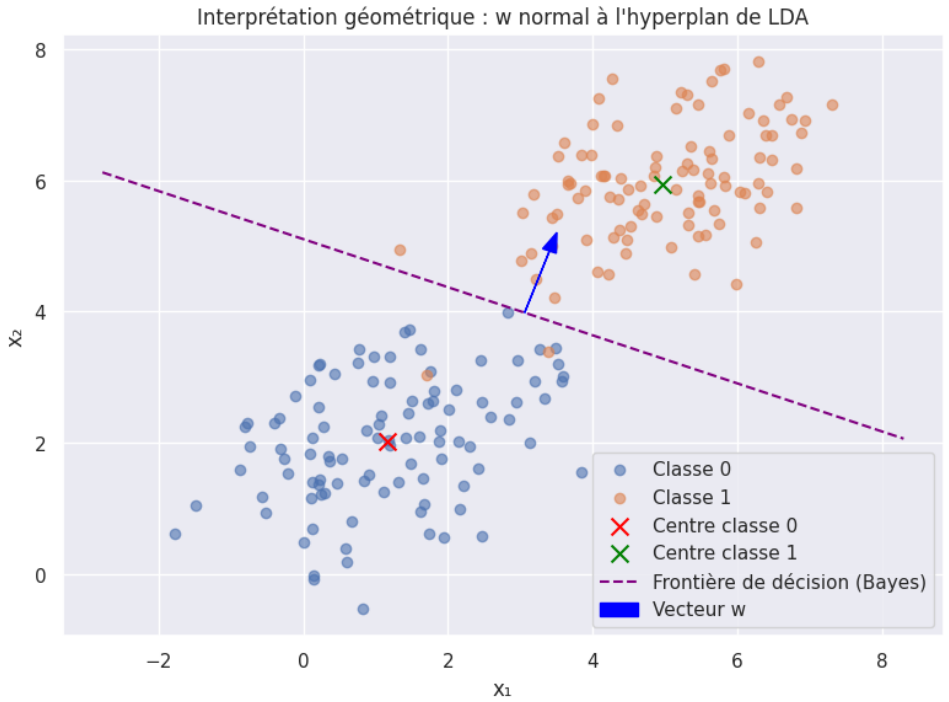
$$\begin{cases} \hat{Y} = 1 & \text{si } \mathbf{x}^\top \mathbf{w} > c \\ \hat{Y} = 0 & \text{si } \mathbf{x}^\top \mathbf{w} < c \end{cases}$$

- $\mathbf{x}^\top \mathbf{w}$ mesure la position relative d'un point par rapport à la frontière définie par l'hyperplan $H = \left\{ \mathbf{x} \in \mathbb{R}^p \;\middle|\; \mathbf{w}^\top \mathbf{x} = c \right\}$.



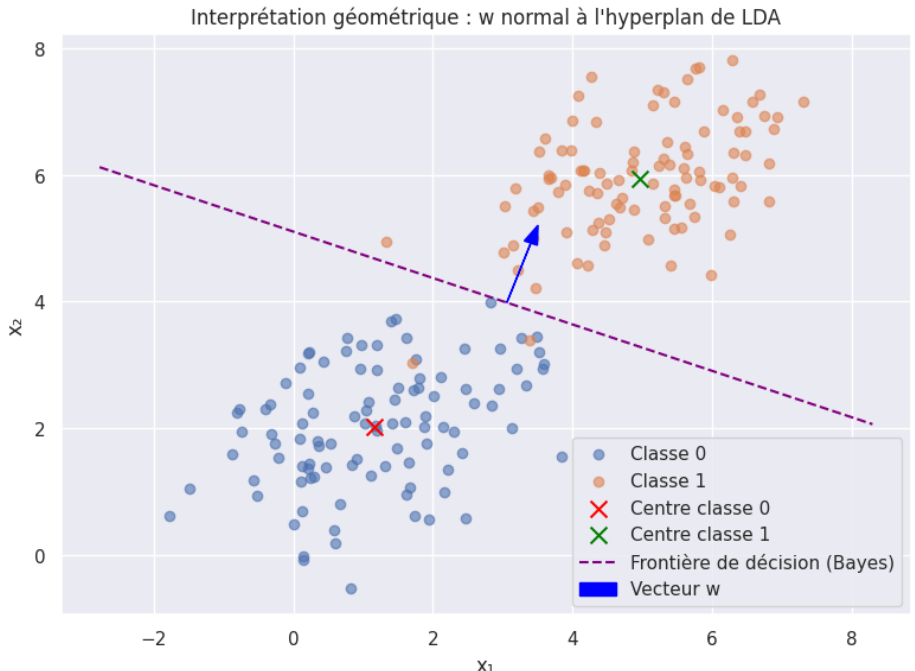
LDA avec Formulation Bayesienne

- $\mathbf{w}$ est dirigé dans une direction entre le centre de la classe 0 $\boldsymbol{\mu}_0$ vers celui de la classe 1 $\boldsymbol{\mu}_1$.
- Cette direction $\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$ est déformée par l'étalement (dispersion) de chacune des classes et elle est exprimée par la covariance commune $\boldsymbol{\Sigma}$.
- Si on se déplace dans la direction de $\mathbf{w}$, on augmente les chances que $\hat{Y} = 1$.



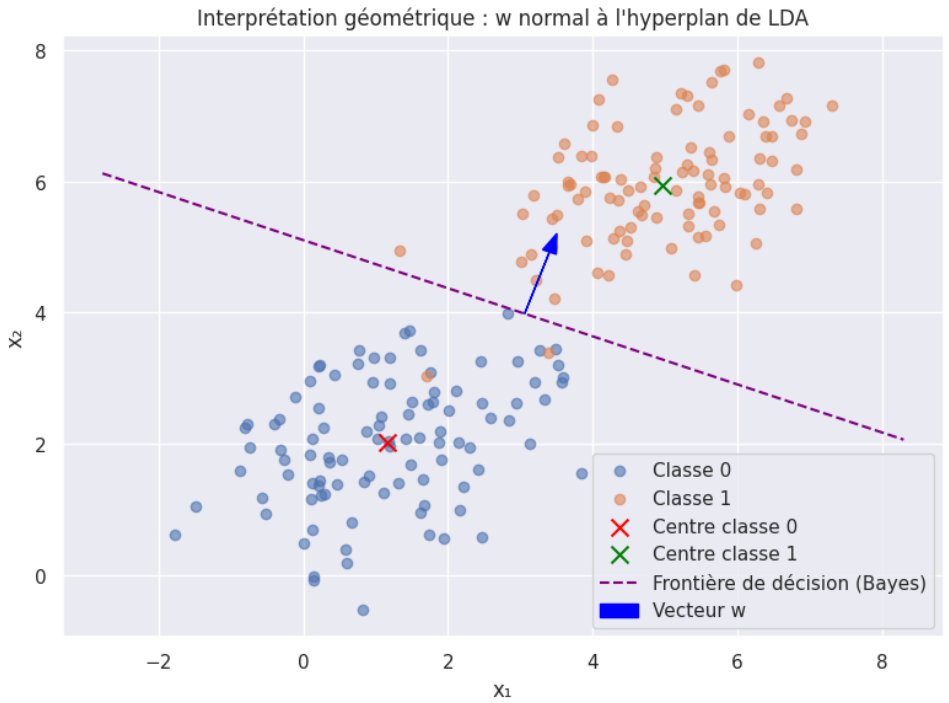
Interprétation du score discriminant

- Le plan de séparation H est :
  - Centré dans l'espace transformé par $\boldsymbol{\Sigma}^{-1}$ entre les deux moyennes $\boldsymbol{\mu}_0$ et $\boldsymbol{\mu}_1$.
  - La matrice de covariance $\boldsymbol{\Sigma}$ permet de tenir compte de la dispersion dans chaque classe.
- Plus $\mathbf{x}^\top \mathbf{w}$ est proche de $\boldsymbol{\mu}_1^\top \mathbf{w}$, plus la probabilité que $\hat{Y} = 1$ est grande.
- Plus il est proche de $\boldsymbol{\mu}_0^\top \mathbf{w}$, plus on pense que $\hat{Y} = 0$.



LDA avec Formulation Bayesienne

Élément	**Interprétation**
$\mathbf{w}$	Direction de séparation entre les classes
$\mathbf{x}^\top \mathbf{w}$	Score projeté utilisé pour la classification
Hyperplan $\mathbf{x}^\top \mathbf{w} = c$	Frontière linéaire orthogonale à $\mathbf{w}$
Sens de $\mathbf{w}$	de $\boldsymbol{\mu}_0$ vers $\boldsymbol{\mu}_1$ dans l'espace transformé par $\boldsymbol{\Sigma}^{-}1$



LDA avec Formulation de Fisher

LDA avec Formulation de Fisher

- Jusqu'ici, nous avons présenté la LDA dans une formulation bayésienne, fondée sur des hypothèses probabilistes (lois normales, même matrice de covariance).
- Il existe une formulation purement géométrique, introduite par Sir Ronald Fisher, qui repose uniquement sur les notions de variance inter-classes et intra-classes.
- Cette formulation
  a été introduite par **Sir Ronald A. Fisher** dans son article fondateur :
  "The Use of Multiple Measurements in Taxonomic Problems"
  R. A. Fisher, *Annals of Eugenics, 1936.*
- Il y introduit un critère de séparation entre classes fondé sur le rapport entre la variance inter-classe et la variance intra-classe.

LDA avec Formulation de Fisher

- Le critère de Fisher est donné par la formule suivante :

$$J(\mathbf{w}) = \frac{\text{Variance inter-classes projetée dans la direction de } \mathbf{w}}{\text{Variance intra-classe projetée dans la direction de } \mathbf{w}}$$
$$= \frac{\text{Var}(\mathbf{S}_B\mathbf{w})}{\text{Var}(\mathbf{S}_W\mathbf{w})} = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}}.$$

  - $\mathbf{S}_B \in \mathbb{R}^{p\times p}$: matrice de dispersion inter-classes (between-class scatter matrix)
$$\mathbf{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top$$
  - $\mathbf{S}_W \in \mathbb{R}^{p\times p}$: matrice de dispersion intra-classes (within-class scatter matrix)
$$\mathbf{S}_W = \underbrace{\sum_{i\in\mathcal{C}_0}(\mathbf{x}_i - \boldsymbol{\mu}_0)(\mathbf{x}_i - \boldsymbol{\mu}_0)^\top}_{\tilde{\mathbf{x}}_{\mathcal{C}_0}^\top \tilde{\mathbf{x}}_{\mathcal{C}_0}} + \underbrace{\sum_{i\in\mathcal{C}_1}(\mathbf{x}_i - \boldsymbol{\mu}_1)(\mathbf{x}_i - \boldsymbol{\mu}_1)^\top}_{\tilde{\mathbf{x}}_{\mathcal{C}_1}^\top \tilde{\mathbf{x}}_{\mathcal{C}_1}}$$
  - Ici, pour chaque classe $k \in \{0, 1\}$:
    - $\tilde{\mathbf{X}}_{\mathcal{C}_k}$ est la matrice centrée des observations de la classe k.
    - Chaque ligne de $\tilde{\mathbf{X}}_{\mathcal{C}_k}$ correspond à $(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top$ pour un $i \in \mathcal{C}_k$.

LDA avec Formulation de Fisher

- Sous l'hypothèse que **les deux classes partagent la même matrice de covariance Σ,** on veut trouver la solution du problème d'optimisation suivant :

$$\arg\max_{\mathbf{w}} \; J(\mathbf{w}) = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}}$$

- Ce rapport mesure :
  - en haut : la variance entre les différentes classes (**Signal : Séparation des classes).**
  - en bas : la variance au sein de chacune des classes (**Bruit : Combien l'information est différente au sein de chaque classe).**
- **Objectif :** Trouver une direction $\mathbf{w}$ dans l'espace des variables explicatives telle que la projection des données $\mathbf{z} = \mathbf{w}^\top \mathbf{x}$ permette une séparation maximale entre les classes.
- On cherche alors à :
  - **Maximiser** la variabilité (dispersion) entre les classes **en maximisant** $\mathbf{w}^\top \mathbf{S}_B \mathbf{w}$.
  - **Minimiser** la variabilité (dispersion) à l'intérieur de chaque classe **en manimisant** $\mathbf{w}^\top \mathbf{S}_W \mathbf{w}$.

LDA avec Formulation de Fisher

- $\mathbf{S}_B$ est la matrice de dispersion **entre les classes** (inter-classes), qui mesure l'écart entre les moyennes des groupes.
- $\mathbf{S}_W$ est la matrice de dispersion **intra-classes**, qui mesure la variabilité des points autour de leur centre de classe.
- Le critère $J(\mathbf{w})$ est un **rapport de formes quadratiques**, structure fréquente en optimisation.
- Le numérateur $\mathbf{w}^\top \mathbf{S}_B \mathbf{w}$ mesure à quel point les classes sont séparées une fois projetées sur la direction $\mathbf{w}$.
- Le dénominateur $\mathbf{w}^\top \mathbf{S}_W \mathbf{w}$ mesure le "bruit projeté", c'est-à-dire la variance résiduelle dans chaque classe.
- En maximisant $J(\mathbf{w})$, on cherche donc une direction $\mathbf{w}$ qui maximise le rapport signal / bruit dans l'espace projeté.
- **Intuition :** on souhaite que les classes soient bien séparées (grand signal) et peu dispersées (peu de bruit).

LDA avec Formulation de Fisher

- Nous voulons maximiser le critère de Fisher:

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}}$$

- Cette forme de quotient est appelée **quotient de Rayleigh** et est courante en optimisation quadratique.
- Elle est invariante à la norme de $\mathbf{w}$, donc on impose une contrainte pour l'optimiser.
- On fixe : $\mathbf{w}^\top \mathbf{S}_W \mathbf{w} = 1$ et on transforme le problème en une maximisation sous contrainte :

$$\max_{\mathbf{w}} \quad \mathbf{w}^\top \mathbf{S}_B \mathbf{w} \quad \text{sous la contrainte} \quad \mathbf{w}^\top \mathbf{S}_W \mathbf{w} = 1$$

- On introduit un multiplicateur de Lagrange λ.
- On définit la fonction de Lagrangien :

$$\mathcal{L}(\mathbf{w}, \lambda) = \mathbf{w}^\top \mathbf{S}_B \mathbf{w} - \lambda(\mathbf{w}^\top \mathbf{S}_W \mathbf{w} - 1)$$

LDA avec Formulation de Fisher

- On dérive $\mathcal{L}$ par rapport à $\mathbf{w}$.
- Rappel utile : si $\mathbf{A}$ est une matrice symétrique, alors :

$$\frac{d}{d\mathbf{w}}(\mathbf{w}^\top \mathbf{A}\mathbf{w}) = 2\mathbf{A}\mathbf{w}$$

- En appliquant cette règle :

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 2\mathbf{S}_B\mathbf{w} - 2\lambda\mathbf{S}_W\mathbf{w}$$

- On annule le gradient pour maximiser :

$$2\mathbf{S}_B\mathbf{w} - 2\lambda\mathbf{S}_W\mathbf{w} = 0 \quad \Longrightarrow \quad \mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}$$

LDA avec Formulation de Fisher

- L'équation obtenue est une **équation aux valeurs propres généralisée** :

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}.$$

- Si $\mathbf{S}_W$ est inversible, on peut écrire :

$$\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{w} = \lambda\mathbf{w}.$$

- La solution optimale $\mathbf{w}^*$ est donc le **vecteur propre principal** (associé à la plus grande valeur propre) de la matrice $\mathbf{S}_W^{-1}\mathbf{S}_B$.

Interprétation :

- $\mathbf{S}_B$ mesure la séparation entre les centres des classes.
- $\mathbf{S}_W$ mesure la dispersion à l'intérieur des classes.
- La direction optimale $\mathbf{w}^*$ maximise le rapport entre ces deux quantités.

- **Dérivée seconde (Hessienne) :** La dérivée seconde du Lagrangien par rapport à $\mathbf{w}$ s'écrit :

$$\nabla^2_{\mathbf{w}}\mathcal{L} = 2\left(\mathbf{S}_B - \lambda \mathbf{S}_W\right).$$

- **Interprétation :** Cette Hessienne est une forme quadratique qui permet d'analyser la nature du point critique obtenu :

$$\nabla_{\mathbf{w}}\mathcal{L} = 0 \quad \Longrightarrow \quad \mathbf{S}_B\mathbf{w} = \lambda \mathbf{S}_W\mathbf{w}.$$

- Le comportement de la Hessienne $\nabla^2_{\mathbf{w}}\mathcal{L}$ est analogue à celui observé dans la PCA :
  - Elle est **négative semi-définie** dans les directions orthogonales à $\mathbf{w}^*$, ce qui indique que le point critique est bien un **maximum local (et global)**.
  - Le critère de Fisher $J(\mathbf{w})$ est donc **concave sur le sous-espace orthogonal** à la direction optimale.
  - Cela signifie que si on se déplace dans une direction orthogonale à la solution optimale $\mathbf{w}^*$, la valeur du critère de Fisher $J(\mathbf{w})$ diminue (ou reste constante). Autrement dit, $\mathbf{w}^*$ est un maximum global du critère, et la fonction est localement concave dans les directions orthogonales à cette solution.

LDA avec Formulation de Fisher

- **Objectif** : Sous l'hypothèse que **les deux classes partagent la même matrice de covariance Σ,** on veut trouver la solution du problème d'optimisation suivant :

$$\arg\max_{\mathbf{w}} \; J(\mathbf{w}) = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}}.$$

- Où :
  - $\mathbf{S}_B$ est la matrice de dispersion entre les classes (Signal).
  - $\mathbf{S}_W$ est la matrice de dispersion intra-classes (bruit).
- Dans le cas de deux classes $Y \in \{0, 1\}$:

$$\mathbf{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top = \mathbf{m}\mathbf{m}^\top$$

- Avec $\mathbf{m} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_0$.
- La matrice $\mathbf{S}_B$ est de rang 1, car elle est construite à partir d'un produit extérieur d'un seul vecteur qui est $\mathbf{m}$.

LDA avec Formulation de Fisher

- Le problème d'optimisation sous contrainte donne :

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}$$

- En remplaçant $\mathbf{S}_B = \mathbf{m}\mathbf{m}^\top$, on obtient :

$$\mathbf{m}\mathbf{m}^\top\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}$$

Étape 3 : Utilisation d'une propriété matricielle

- On utilise l'identité :

$$\mathbf{m}\mathbf{m}^\top \mathbf{w} = (\mathbf{m}^\top \mathbf{w})\mathbf{m}$$

- Ce qui donne :

$$(\mathbf{m}^\top \mathbf{w})\mathbf{m} = \lambda \mathbf{S}_W \mathbf{w}$$

$$\Rightarrow \frac{\mathbf{m}^\top \mathbf{w}}{\lambda}\mathbf{m} = \mathbf{S}_W \mathbf{w}$$

- Cette équation signifie que $\mathbf{S}_W \mathbf{w}$ est colinéaire à $\mathbf{m}$.
- Ainsi :

$$\mathbf{S}_W \mathbf{w} \propto \mathbf{m} \quad \Rightarrow \quad \mathbf{w} \propto \mathbf{S}_W^{-1}\mathbf{m}$$

- Donc, la solution optimale est :

$$\boxed{\mathbf{w}^\star = \mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}$$

- Le vecteur $\mathbf{w}^\star$ pointe dans la direction entre les centres des classes, dans l'espace transformé par l'inverse la matrice de covariance intra-classe $\mathbf{S}_W^{-1}$.

LDA avec Formulation de Fisher

- La solution optimale est :

$$\boxed{\mathbf{w}^\star = \mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}$$

- Le vecteur $\mathbf{w}^\star$ pointe dans la direction entre les centres des classes, dans l'espace transformé par l'inverse la matrice de covariance intra-classe $\mathbf{S}_W^{-1}$.
- On projette chaque observation $\mathbf{x}_i$:

$$z_i = \mathbf{w}^{\star\top}\mathbf{x}_i$$

- On choisit ensuite un seuil optimal pour la classification, par exemple à mi-chemin entre :

$$\mathbf{w}^{\star\top}\boldsymbol{\mu}_0 \quad \text{et} \quad \mathbf{w}^{\star\top}\boldsymbol{\mu}_1$$

- $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ est le vecteur qui relie les centres de gravité des deux classes ;
- La matrice $\mathbf{S}_W^{-1}$ agit comme une correction qui pondère cette direction selon la variabilité des dimensions (moins une variable est fiable, moins elle est utilisée).

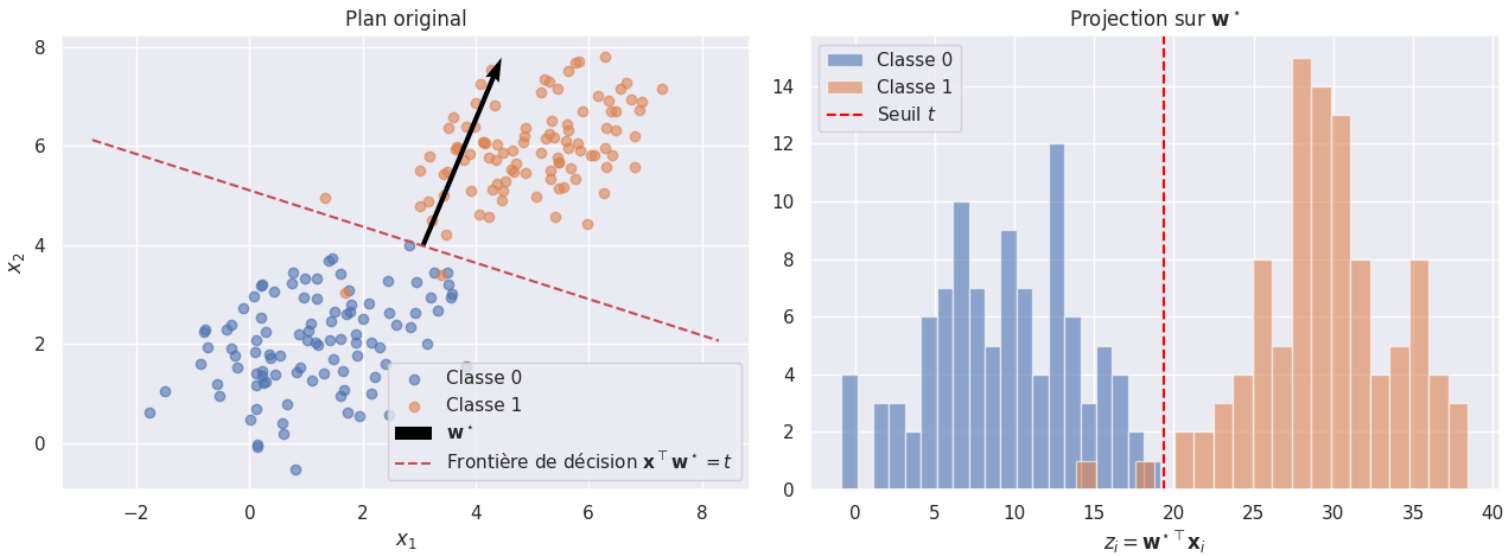
Comparaison du seuil de décision : Fisher vs Bayesien

- Contrairement à l'approche bayésienne, la formulation de Fisher ne détermine pas directement une constante c telle que :

$$\mathbf{x}^\top \mathbf{w} = c$$

- Mais une fois les données projetées sur la direction optimale $\mathbf{w}^\star$, la classification se fait en comparant les scores projetés :

$$z_i = \mathbf{w}^{\star\top} \mathbf{x}_i \quad \Rightarrow \quad \widehat{Y}_i = \begin{cases} 1 & \text{si } z_i > t \\ 0 & \text{sinon} \end{cases}$$

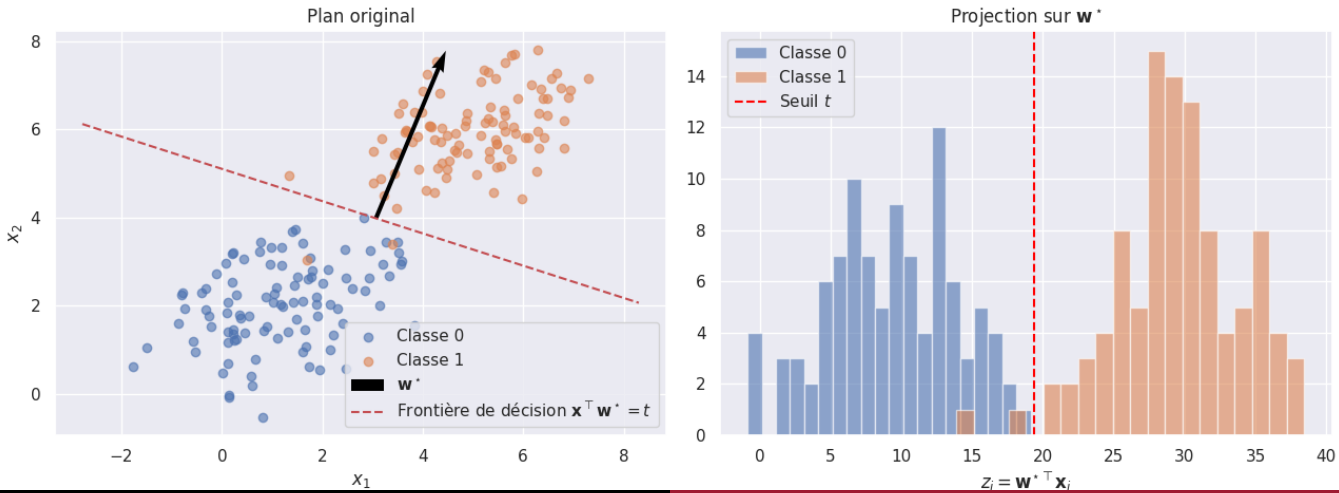


Comparaison du seuil de décision : Fisher vs Bayesien

- **Comment fixer ce seuil t ?**
  - En pratique, on choisit souvent :

$$t = \frac{1}{2}\left(\mathbf{w}^{\star\top}\boldsymbol{\mu}_0 + \mathbf{w}^{\star\top}\boldsymbol{\mu}_1\right)$$

  - Il s'agit du milieu entre les centres projetés des deux classes. Ce choix approxime la frontière bayésienne si les classes sont équilibrées.
- **Bayesien :** le seuil c est calculé analytiquement à partir des densités de probabilité : $c = \frac{1}{2}\left(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0\right) - \log\left(\frac{p_1}{p_0}\right)$
- **Fisher :** le seuil t est choisi empiriquement comme moyenne des centres projetés : $t = \frac{1}{2}\left(\mathbf{w}^\top\boldsymbol{\mu}_0 + \mathbf{w}^\top\boldsymbol{\mu}_1\right)$

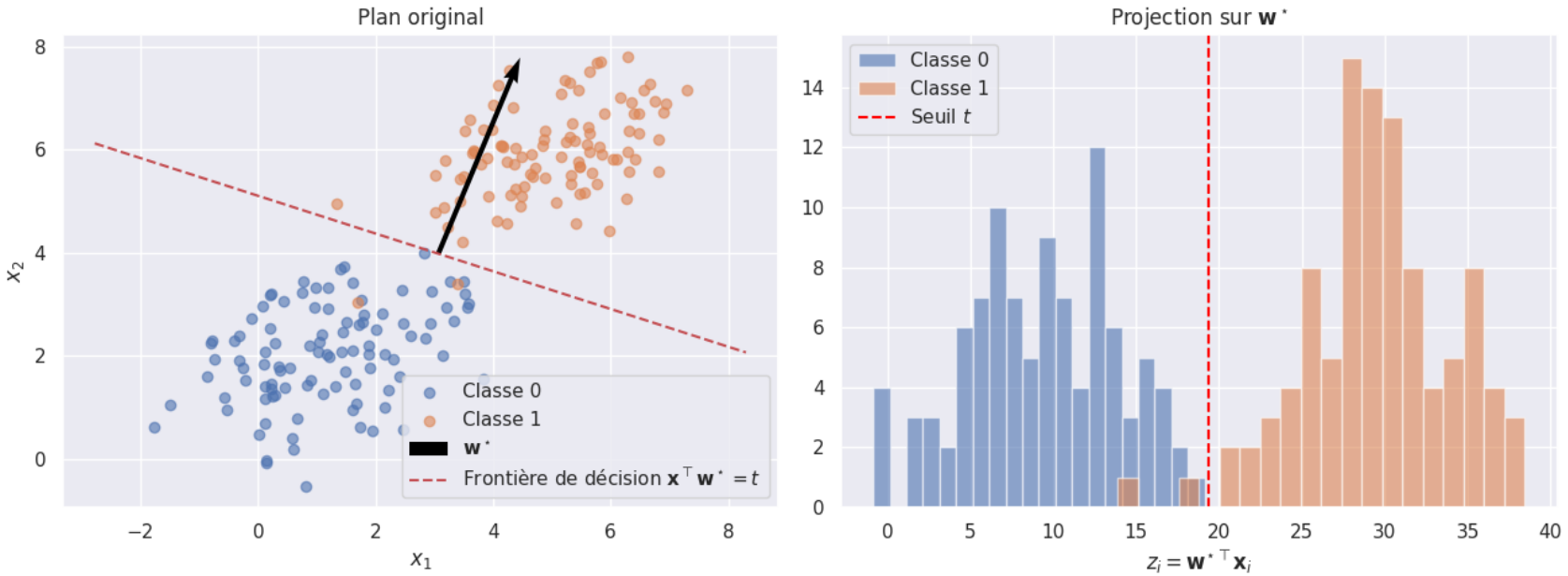


Comparaison des seuils : Bayesien vs Fisher

- **Seuil dans l'approche bayésienne (classes équilibrées)** :
  - Lorsque $p_0 = p_1$, le seuil analytique pour la frontière de décision est :

$$c = \frac{1}{2}\left(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0\right)$$

  - Cette constante c définit l'hyperplan : $\mathbf{x}^\top \mathbf{w} = c$, avec $\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$.



Comparaison des seuils : Bayesien vs Fisher

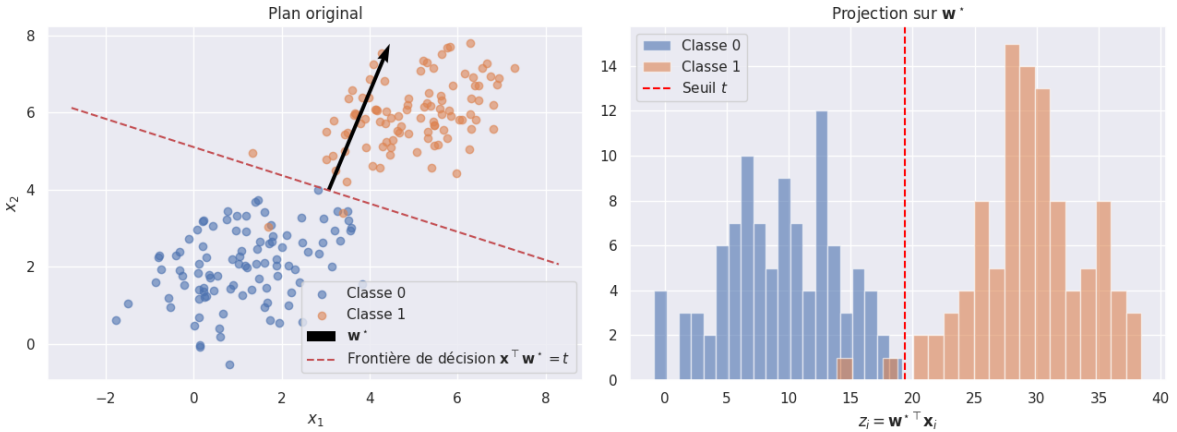
- **Seuil dans l'approche de Fisher (après projection)** :
  - On choisit un seuil :

$$t = \frac{1}{2}\left(\mathbf{w}^\top \boldsymbol{\mu}_0 + \mathbf{w}^\top \boldsymbol{\mu}_1\right)$$

  - En remplaçant $\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$, on obtient :

$$\begin{aligned} t &= \frac{1}{2}\left((\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_0 + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1\right) \\ &= \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1) \end{aligned}$$

- **Conclusion** : Ce développement montre que $\boxed{t = c}$ lorsque $p_0 = p_1$, càd si les classes sont équilibrées.

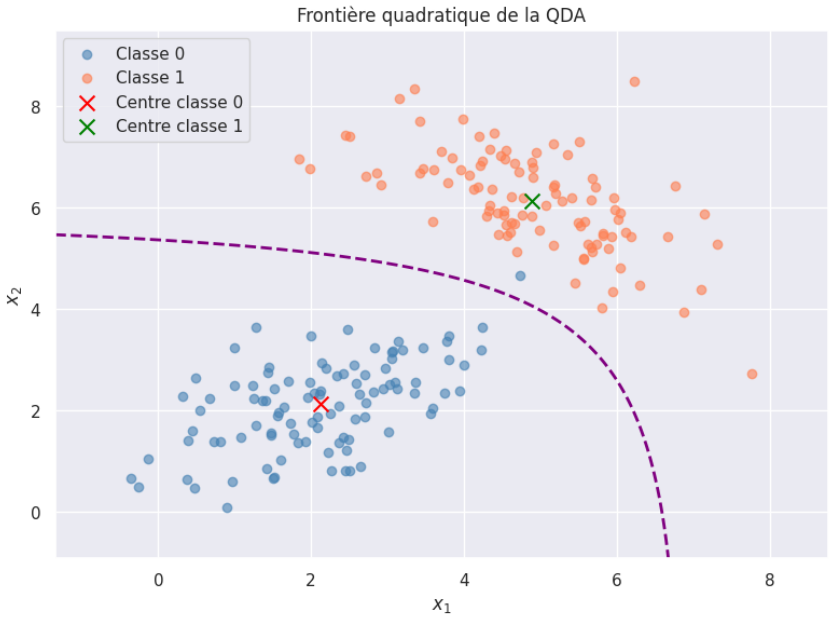


Note : Quadratic Discriminant Analysis (QDA)

- **L'Analyse Discriminante Quadratique** (*Quadratic Discriminant Analysis* **QDA**) est une généralisation de la LDA.
- Contrairement à la LDA, la QDA **n'impose pas l'hypothèse d'homoscédasticité** où l'on suppose l'égalité des matrices de covariance pour toutes les classes :

$$\mathbf{X}_i \mid Y_i = y \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$$

- Chaque classe possède sa propre matrice de covariance $\boldsymbol{\Sigma}_y$.



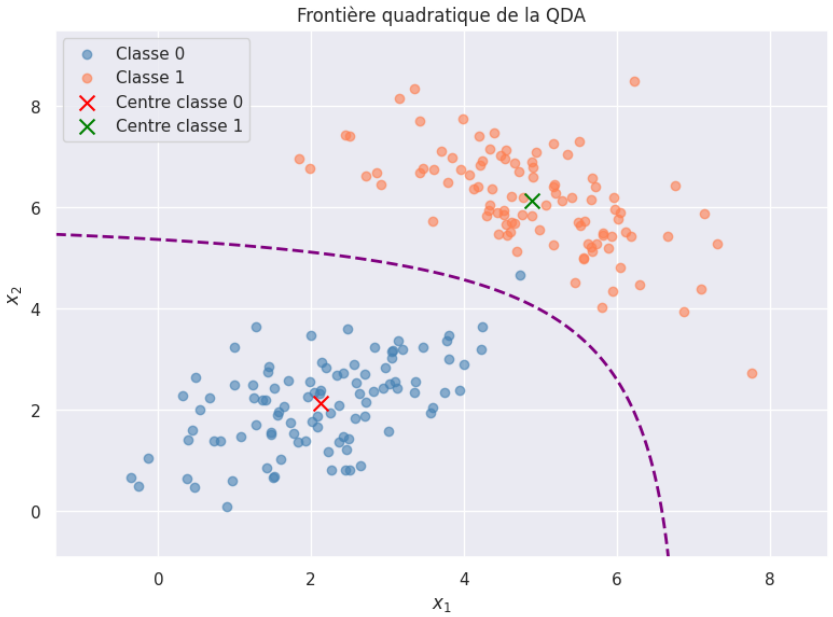
Note : Quadratic Discriminant Analysis (QDA)

- Chaque classe possède sa propre matrice de covariance $\boldsymbol{\Sigma}_y$.
- La fonction discriminante pour chaque classe $y \in \{0, 1\}$ est alors donnée par :

$$\delta_y(\mathbf{x}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_y| - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^\top \boldsymbol{\Sigma}_y^{-1}(\mathbf{x} - \boldsymbol{\mu}_y) + \log(p_y)$$

- La frontière de décision est obtenue en égalisant ces fonctions :

$$\delta_1(\mathbf{x}) = \delta_0(\mathbf{x}).$$



Quand utiliser la QDA ?

- La fonction discriminante pour chaque classe $y \in \{0, 1\}$ est alors donnée par :

$$\delta_y(\mathbf{x}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_y| - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^\top \boldsymbol{\Sigma}_y^{-1}(\mathbf{x} - \boldsymbol{\mu}_y) + \log(p_y)$$

- La frontière de décision est obtenue en égalisant ces fonctions :

$$\delta_1(\mathbf{x}) = \delta_0(\mathbf{x}).$$

- Ce qui donne une équation du second degré :

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c = 0$$

- Avec :

$$\mathbf{A} = \frac{1}{2}\left(\boldsymbol{\Sigma}_0^{-1} - \boldsymbol{\Sigma}_1^{-1}\right)$$

$$\mathbf{b} = \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0$$

$$c = \log\left(\frac{p_1}{p_0}\right) - \frac{1}{2}\log\left(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}\right) - \frac{1}{2}\left(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right)$$

Quand utiliser la QDA ?

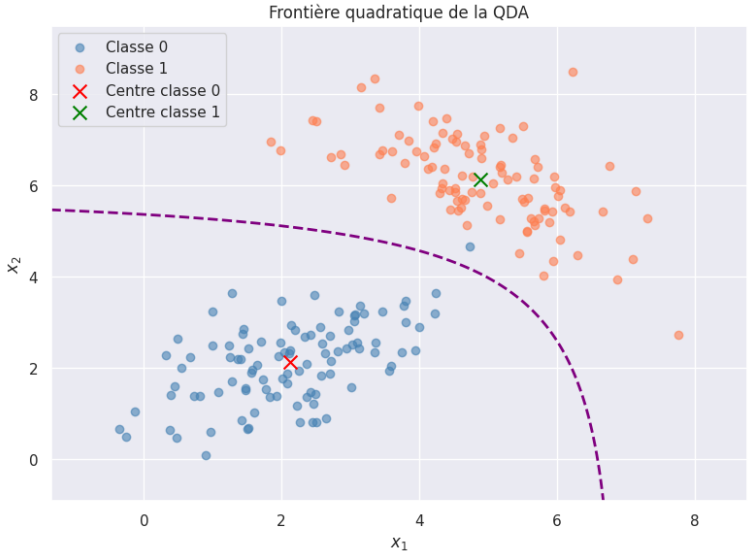
- La fonction discriminante pour chaque classe $y \in \{0, 1\}$ est :

$$\delta_y(\mathbf{x}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_y| - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_y)^\top \boldsymbol{\Sigma}_y^{-1}(\mathbf{x} - \boldsymbol{\mu}_y) + \log(p_y)$$

- La frontière de décision est obtenue en égalisant ces fonctions :

$$\delta_1(\mathbf{x}) = \delta_0(\mathbf{x}).$$

- La QDA est utile lorsque l'hypothèse d'homoscédasticité est clairement violée.
- Elle l'est aussi si chaque classe a une forme de distribution différente.



Machine à Vecteurs de Support (SVMs)

Machine à Vecteurs de Support (SVMs)

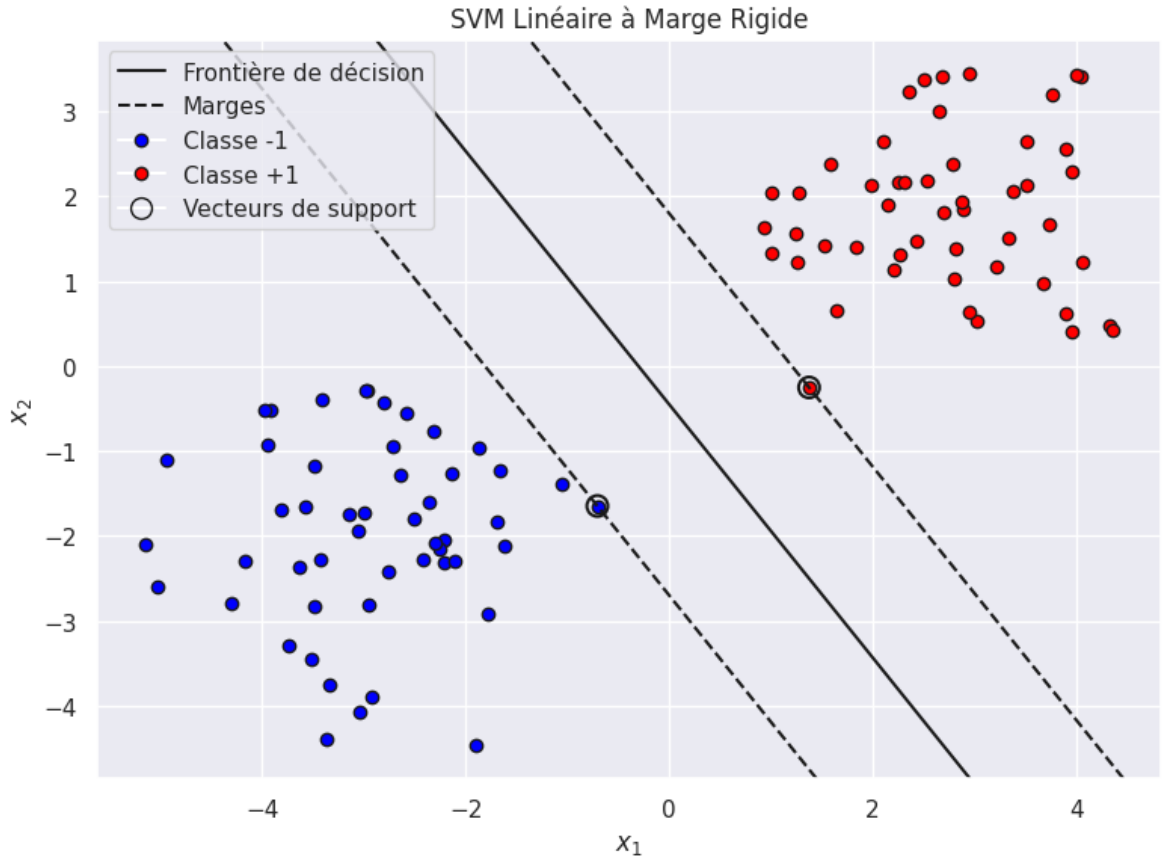
- SVM propose une nouvelle perspective :
  "Et si, au lieu d'assumer des distributions, on apprenait simplement l'hyperplan qui maximise la séparation géométrique entre les classes ?"

Machine à Vecteurs de Support (SVMs)

- **Motivation** : pourquoi aller au-delà de LDA/QDA ?
  - Les méthodes LDA et QDA reposent sur des hypothèses probabilistes (normalité des classes, homoscédasticité pour LDA).
  - Ces hypothèses peuvent être trop fortes ou irréalistes dans de nombreuses situations réelles.
  - On souhaite une méthode plus souple, **moins dépendante de la distribution des données**.
- **Nature du modèle** : les **SVMs** (Support Vector Machines) sont des modèles **discriminatifs**.
  - Ils cherchent directement une **frontière de séparation optimale** entre les classes.
  - Contrairement à LDA/QDA, ils ne modélisent pas la distribution $P(\mathbf{X} \mid Y)$.

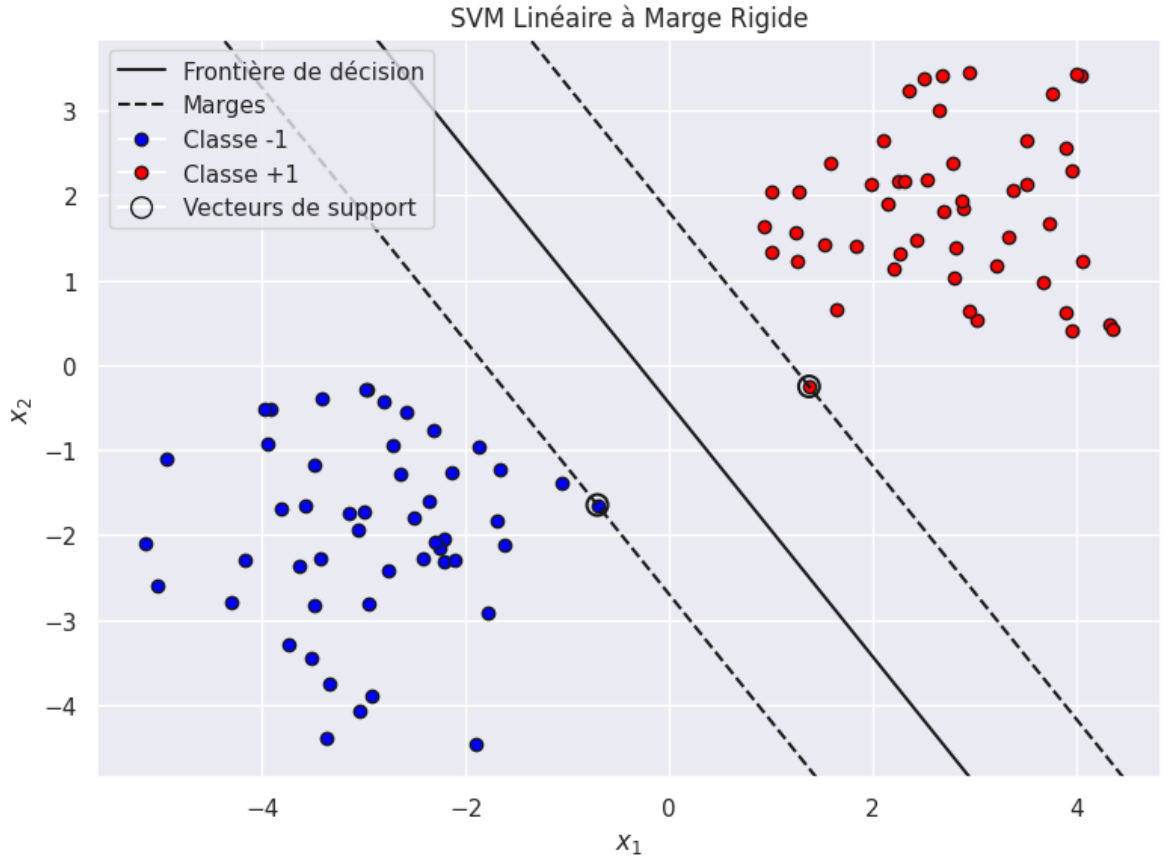
Machine à Vecteurs de Support (SVMs)

- **Comparaison avec LDA** :
  - **LDA** : maximise le rapport de variances (inter-classes vs intra-classe), ce qui donne une séparation **analytique**.
  - **SVM** : maximise la **marge géométrique** (espace de séparation) entre les classes, sans hypothèse de distribution.



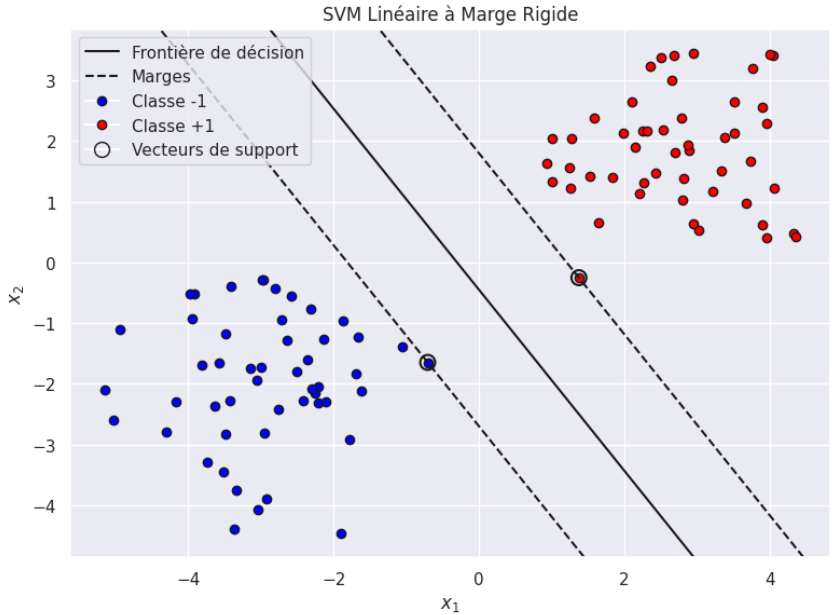
Machine à Vecteurs de Support (SVMs)

- **Cas 2D linéairement séparable** :
  - L'idée est de trouver l'hyperplan séparant les classes avec la plus grande marge (espace de séparation).
  - Seuls les points proches de la frontière (appelés *vecteurs de support*) déterminent la solution.



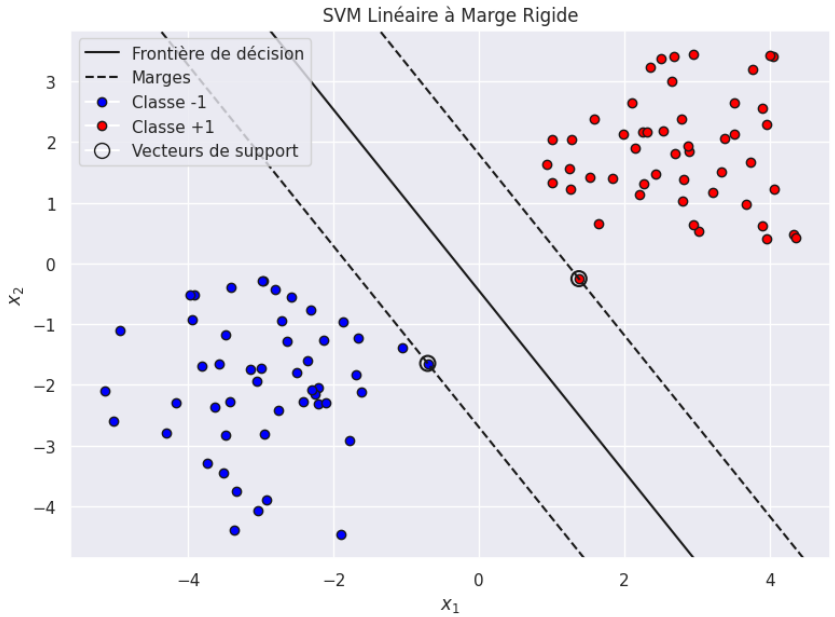
Notions clés illustrées dans le graphique SVM

- **Frontière de décision** (ligne noire pleine) :
  - Équation : $\mathbf{w}^\top \mathbf{x} + b = 0$;
  - C'est l'hyperplan séparant les deux classes.
- **Marges** (lignes noires pointillées) :
  - Équations : $\mathbf{w}^\top \mathbf{x} + b = \pm 1$;
  - Bandes définissant la zone de tolérance autour de la frontière.



Machine à Vecteurs de Support (SVMs)

- **Vecteurs de support** (cerclés en noir) :
  - Points les plus proches de la frontière ;
  - Ils satisfont $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1$;
  - Essentiels pour définir la solution du SVM.
- **Classes** :
  - Classe bleue ($y = -1$) : points en bas à gauche ;
  - Classe rouge ($y = +1$) : points en haut à droite ;
  - L'objectif est de séparer ces classes avec une marge maximale.



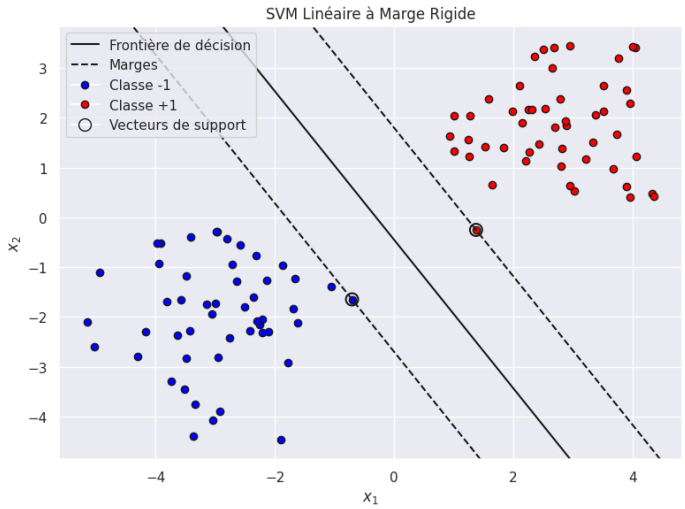
Machine à Vecteurs de Support (SVMs)

- **Objectif du SVM** : Apprendre une frontière de décision linéaire qui sépare deux classes de manière optimale, en maximisant la marge entre elles.
- **Contexte** : On dispose d'un ensemble d'exemples d'entraînement

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^n \quad \text{avec } \mathbf{x}_i \in \mathbb{R}^p, \quad y_i \in \{-1, +1\}$$

- **La Fonction de décision linéaire** est donnée par :

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

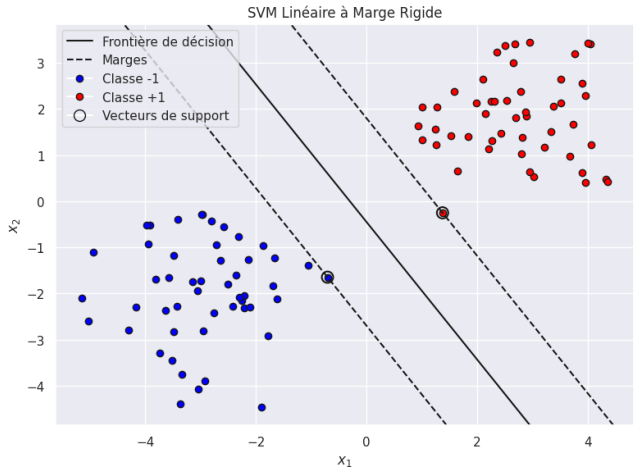


Machine à Vecteurs de Support (SVMs)

- **Pour effectuer la prédiction** on utilise la fonction signe (sign) :

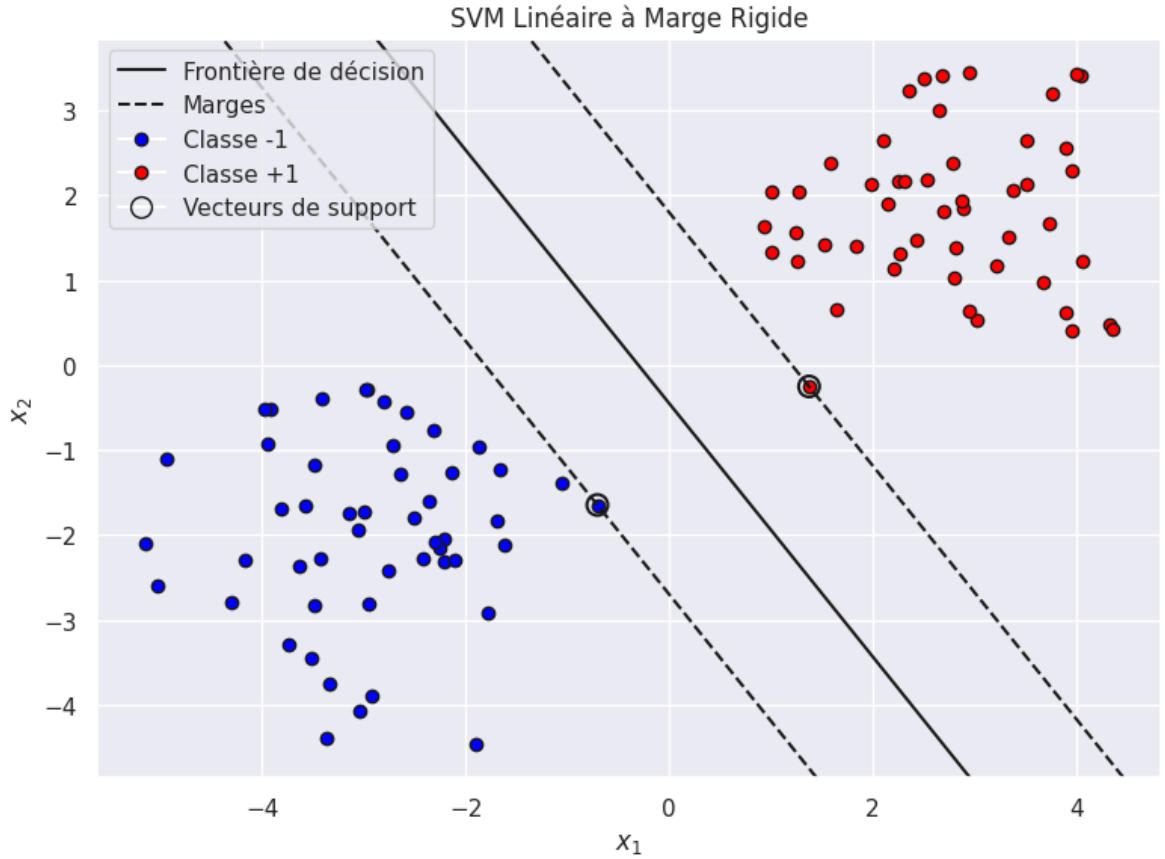
$$\widehat{y}_i = \text{sign}(f(\mathbf{x}_i)) = \text{sign}(\mathbf{w}^\top \mathbf{x}_i + b)$$

- **Interprétation de la quantité** $y_i(\mathbf{w}^\top \mathbf{x}_i + b)$:
  - $y_i \in \{-1, +1\}$ est le label réel.
  - $\mathbf{w}^\top \mathbf{x}_i + b$ est la sortie (output) du modèle pour l'observation $\mathbf{x}_i$.
  - Cette quantité $y_i(\mathbf{w}^\top \mathbf{x}_i + b)$ est appelée la **marge fonctionnelle** : elle mesure la confiance du modèle à correctement classifier $\mathbf{x}_i$.
  - Plus $y_i(\mathbf{w}^\top \mathbf{x}_i + b)$ est grande et positive, plus le point est bien classé et éloigné de la frontière.



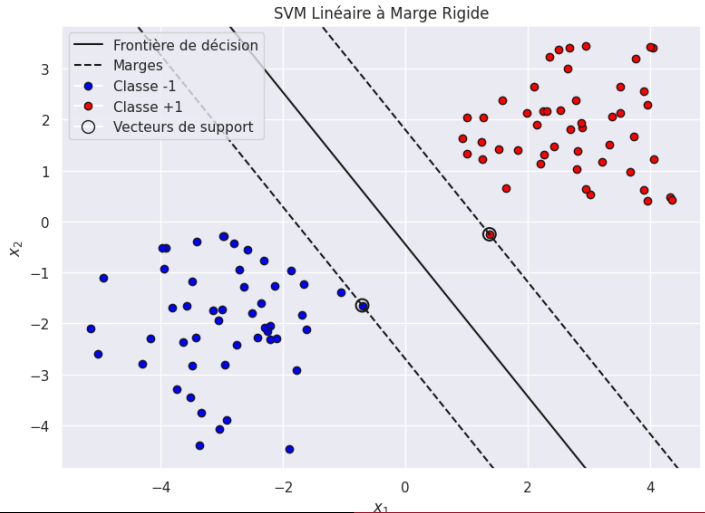
Machine à Vecteurs de Support (SVMs)

- **Utilité de la marge fonctionnelle :**
  - Si $y_i = 1$, on veut que $\mathbf{w}^\top \mathbf{x}_i + b > 0$.
  - Si $y_i = -1$, on veut que $\mathbf{w}^\top \mathbf{x}_i + b < 0$.
  - Donc, dans les deux cas, **on veut que** $y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 0$**.**
  - Et dans un **SVM à marge rigide,** on impose : $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$.



Machine à Vecteurs de Support (SVMs)

- **Cas 1 :** $y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 1 \Rightarrow$ point correctement classé, loin de la frontière, n'influence pas la solution.
- **Cas 2 :** $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1 \Rightarrow$ **point sur la marge, vecteur de support.**
- **Cas 3 :** $0 < y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 1 \Rightarrow$ point du bon côté, mais à l'intérieur de la marge.
- **Cas 4 :** $y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 0 \Rightarrow$ point sur la frontière de décision. Il se peut que le point soit classé incorrectement.
- **Cas 5 :** $y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 0 \Rightarrow$ point mal classé (grande pénalité dans la hinge loss).

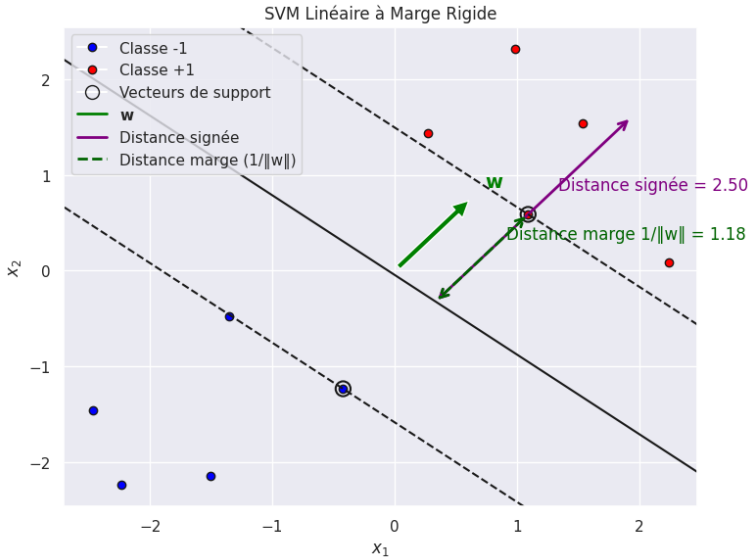


Machine à Vecteurs de Support (SVMs)

- **Représentation de l'hyperplan**
  - Un hyperplan en dimension p est défini par :

$$H = \left\{ \mathbf{x} \in \mathbb{R}^p \;\middle|\; \mathbf{w}^\top \mathbf{x} + b = 0 \right\}, \quad \text{où :}$$

  - $\mathbf{w}$ est un vecteur normal à l'hyperplan.
  - b détermine le décalage (biais) par rapport à l'origine.



Machine à Vecteurs de Support (SVMs)

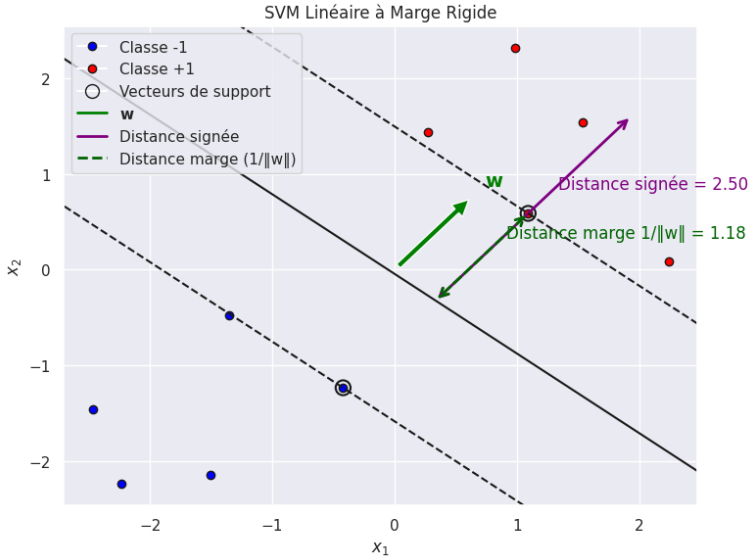
- **Distance d'un point à un hyperplan**
  - Pour un point $\mathbf{x}_i$, la distance euclidienne signée à l'hyperplan est :

$$\text{distance signée} = \frac{\mathbf{w}^\top \mathbf{x}_i + b}{\|\mathbf{w}\|}$$

  - Cela provient de la formule classique en géométrie analytique :

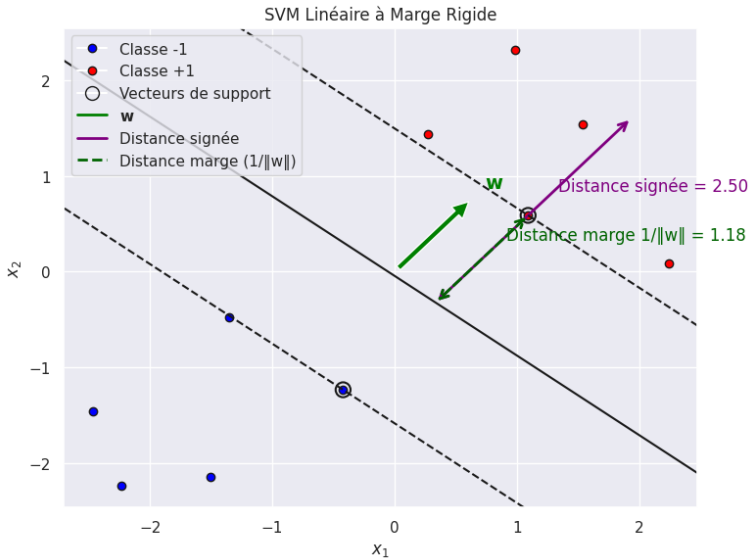
$$\text{distance} = \frac{|ax + by + cz + d|}{\sqrt{a^2 + b^2 + c^2}}$$

  - Ici, $\mathbf{w} = (a, b, c, \dots)$ et $\mathbf{x}$ est un point quelconque.



Machine à Vecteurs de Support (SVMs)

- **L'utilité de la distance signée**
  - Le signe de $\mathbf{w}^\top \mathbf{x}_i + b$ nous indique :
    - $\mathbf{w}^\top \mathbf{x}_i + b > 0$: $\mathbf{x}_i$ est du côté positif de l'hyperplan (même côté que $\mathbf{w}$) ;
    - $\mathbf{w}^\top \mathbf{x}_i + b < 0$: $\mathbf{x}_i$ est du côté opposé ;
    - $\mathbf{w}^\top \mathbf{x}_i + b = 0$: $\mathbf{x}_i$ est sur l'hyperplan.
  - La distance est donc **signée** : elle encode à la fois la direction et la distance.

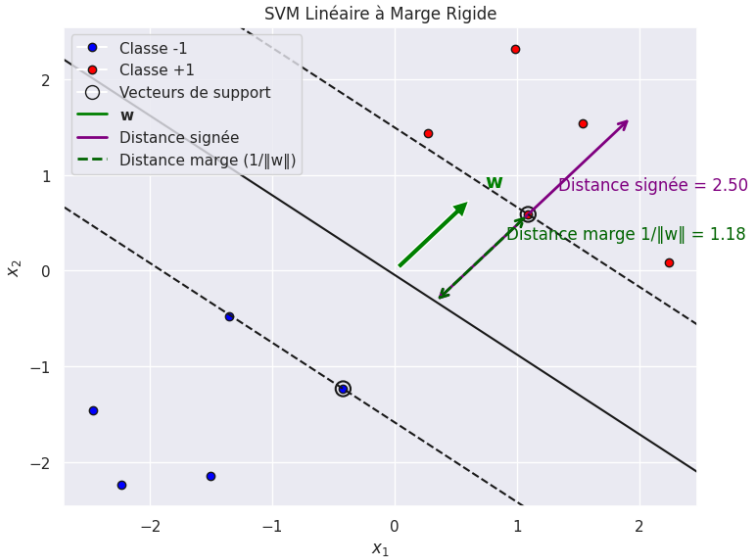


Machine à Vecteurs de Support (SVMs)

- La distance signée est :

$$\frac{\mathbf{w}^\top \mathbf{x}_i + b}{\|\mathbf{w}\|}$$

- Elle permet de quantifier à quel point une observation est loin (ou proche) de la frontière ;
- Elle est au cœur de l'objectif du SVM : **maximiser cette distance minimale** pour les vecteurs de support.



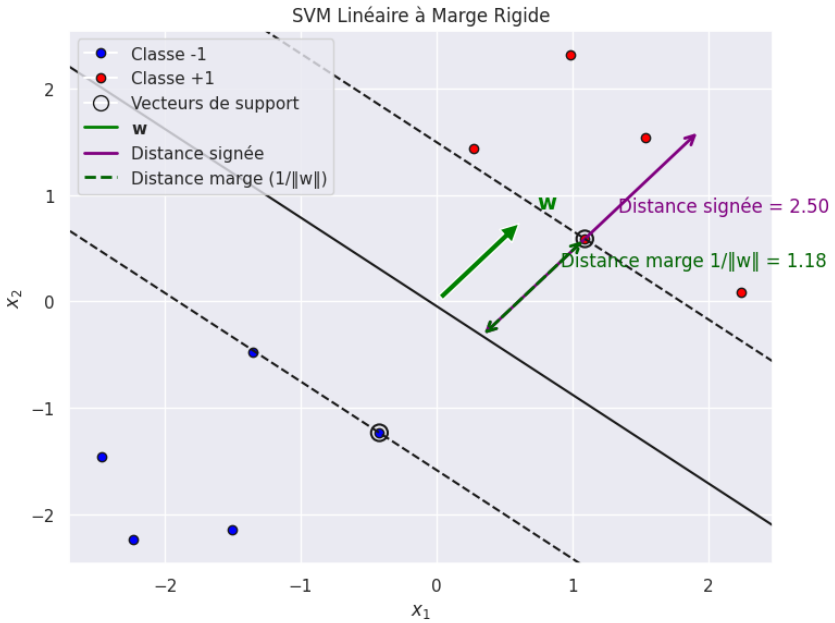
Machine à Vecteurs de Support (SVMs)

- La distance **signée** entre un point $\mathbf{x}_i$ et la frontière de décision est donnée par :

$$\text{distance signée} = \frac{\mathbf{w}^\top \mathbf{x}_i + b}{\|\mathbf{w}\|}$$

- En particulier, pour un point $\mathbf{x}_+$ situé sur la **marge positive** :

$$\mathbf{w}^\top \mathbf{x}_+ + b = 1 \Rightarrow \text{distance à la frontière} = \frac{|\mathbf{w}^\top \mathbf{x}_+ + b - 0|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

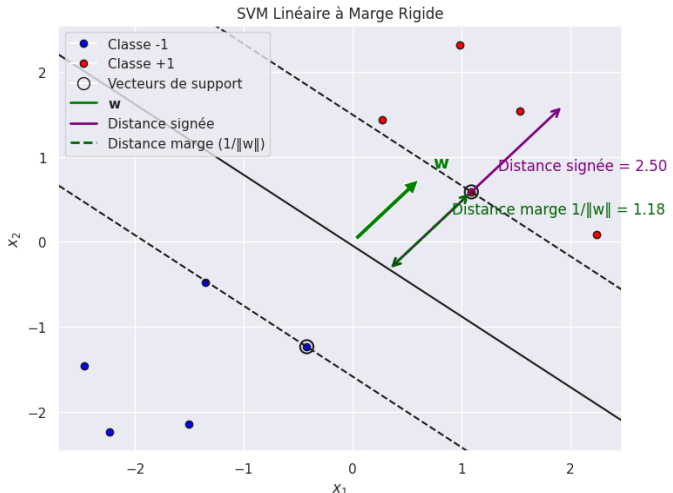


Machine à Vecteurs de Support (SVMs)

- **Pour les vecteurs de support**, la distance qui les sépare de la frontière de décision est la plus courte possible parmi les points de leur classe :

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1 \Rightarrow \text{distance minimale} = \frac{1}{\|\mathbf{w}\|}$$

- **Objectif du SVM** :
  - Trouver l'hyperplan $\mathbf{w}^\top \mathbf{x} + b = 0$ qui maximise cette distance minimale.
  - Autrement dit : **maximiser la marge géométrique** (maximiser la distance minimale $\frac{1}{\|\mathbf{w}\|}$.



Machine à Vecteurs de Support (SVMs)

- L'objectif du SVM est alors de **maximiser la marge géométrique** $\frac{1}{\|\mathbf{w}\|}$.
- Pour des raisons de simplicité mathématique (notamment pour obtenir une fonction objective convexe et différentiable), on choisit de manière équivalente à **minimiser** $\frac{1}{2}\|\mathbf{w}\|^2$.
- Cette minimisation est effectuée **sous contrainte**, en imposant que **tous les exemples d'apprentissage soient correctement classés et en dehors de la marge** :

$$\forall i, \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1.$$

- **La Formulation finale (SVM à marge rigide)** est la suivante :

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{sous la contrainte} \quad \forall i, \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1.$$

Machine à Vecteurs de Support (SVMs)

- À partir du problème primal :

$$\min_{\mathbf{w},b} \ \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{sous la contrainte} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$$

- On forme la **Lagrangienne** avec des multiplicateurs de Lagrange $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \geq 0$:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1]$$

- On minimise $\mathcal{L}$ par rapport à $\mathbf{w}$ et b, ce qui donne :

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0$$

Machine à Vecteurs de Support (SVMs)

- **Problème primal (marge rigide)** :

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{sous la contrainte} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad \forall i$$

- On introduit un multiplicateur de Lagrange $\alpha_i \geq 0$ pour chaque contrainte pour obtenir la formule de la **Lagrangienne associée** :

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1\right]$$

- **Gradient par rapport à $\mathbf{w}$** :

$$\frac{\partial}{\partial \mathbf{w}} \left(\frac{1}{2}\|\mathbf{w}\|^2\right) = \frac{\partial}{\partial \mathbf{w}} \frac{1}{2} \mathbf{w}^\top \mathbf{w} = \frac{2}{2}\mathbf{w} = \mathbf{w}.$$

$$\frac{\partial}{\partial \mathbf{w}} \left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{w}^\top \mathbf{x}_i\right) = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i.$$

Machine à Vecteurs de Support (SVMs)

- Donc :

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i.$$

- En annulant le gradient :

$$\nabla_{\mathbf{w}}\mathcal{L} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i.$$

- **Dérivée par rapport à** b :

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i.$$

- En annulant cette dérivée :

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

Machine à Vecteurs de Support (SVMs)

- On rappelle la Lagrangienne :

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1\right]$$

- Grâce aux conditions du point selle qu'on vient d'obtenir, on sait que $\mathbf{w} = \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j$ et $\sum_{i=1}^{n} \alpha_i y_i = 0.$
- On commence par calculer la norme au carré de $\mathbf{w}$:

$$\|\mathbf{w}\|^2 = \left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right)^\top \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j.$$

- De plus, le terme contenant b disparaît grâce à la contrainte :

$$\sum_{i=1}^{n} \alpha_i y_i b = b \sum_{i=1}^{n} \alpha_i y_i = b \,.\, 0 = 0$$

Machine à Vecteurs de Support (SVMs)

- En remplaçant dans la Lagrangienne, on obtient une nouvelle fonction qui ne dépend que des α_i :

$$\mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j - \sum_{i=1}^{n}\alpha_i y_i \left(\sum_{j=1}^{n}\alpha_j y_j \mathbf{x}_j^\top \mathbf{x}_i\right) + \sum_{i=1}^{n}\alpha_i.$$

- Les deux premiers termes sont égaux, donc on obtient :

$$\mathcal{L}(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j + \sum_{i=1}^{n}\alpha_i.$$

- Il s'agit maintenant d'un problème de maximisation :

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j.$$

sous les contraintes :

$$\alpha_i \geq 0 \quad \forall i, \quad \text{et} \quad \sum_{i=1}^{n}\alpha_i y_i = 0.$$

Machine à Vecteurs de Support (SVMs)

- On vient d'obtenir l'expression du **problème dual** en insérant dans $\mathcal{L}$ les conditions du point selle qu'on avait obtenu précédemment par l'annulation du gradient de $\mathcal{L}$ par rapport à $\mathbf{w}$ et b :

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j.$$

  sous les contraintes :

$$\alpha_i \geq 0 \quad \forall i, \quad \text{et} \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

- Chaque coefficient α_i reflète l'**importance** de l'exemple $(\mathbf{x}_i, y_i)$ dans la construction de l'hyperplan.
  - Si $\alpha_i = 0$, le point $\mathbf{x}_i$ n'est pas un vecteur de support.
  - Si $\alpha_i > 0$, alors $\mathbf{x}_i$ est un **vecteur de support**, c'est-à-dire qu'il influence la solution.

Machine à Vecteurs de Support (SVMs)

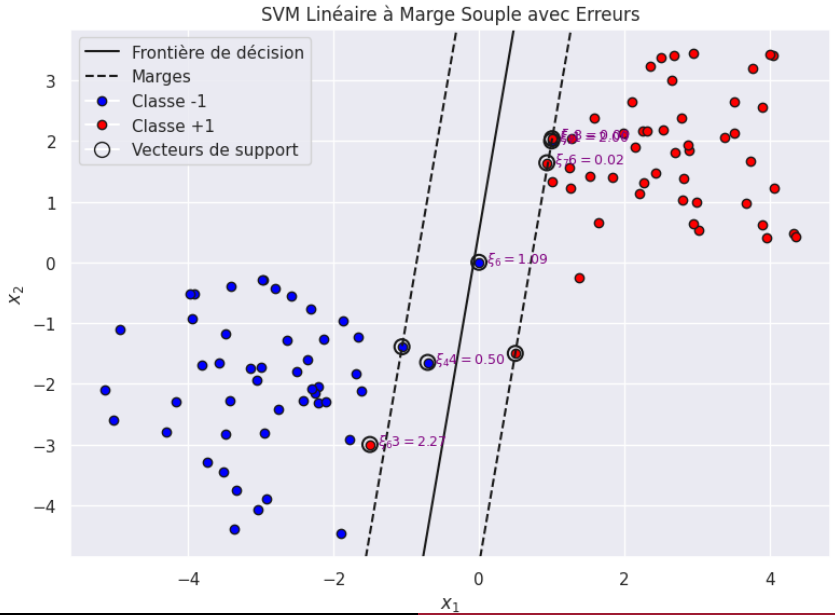
- Le vecteur normal à la frontière est une combinaison linéaire des vecteurs de support :

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i.$$

- Le terme $-\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$ capture la cohérence géométrique entre les exemples :
  - Il favorise une solution où les vecteurs de support sont bien répartis dans l'espace.
  - Un alignement trop forte entre les vecteurs de support ($\mathbf{x}_i^\top \mathbf{x}_j$) augmente ce terme et donc est défavorable.

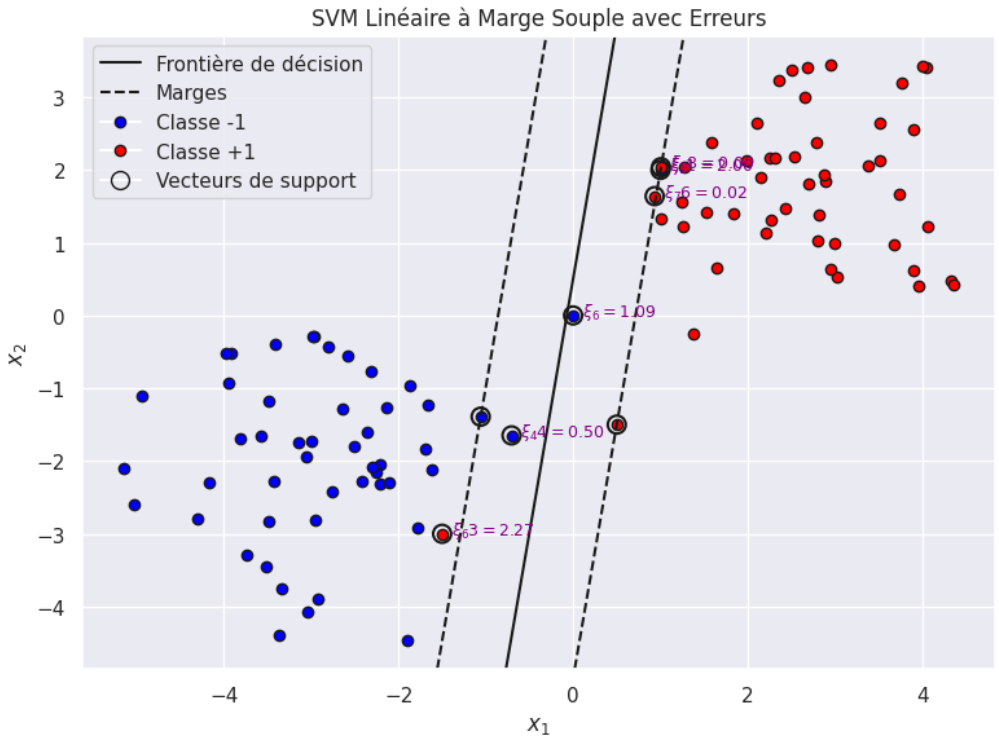
Machine à Vecteurs de Support (SVMs)

- Jusqu'à présent, nous avons supposé que les données sont parfaitement séparables par une frontière linéaire avec une marge non nulle.
- **Mais dans la réalité**, les données :
  - Contiennent du bruit, des erreurs de labellisation (étiquetage).
  - Peuvent se recouvrir partiellement (classes non séparables linéairement).
  - Comportent des outliers qui rendent la séparation rigide impossible ou peu robuste.



Machine à Vecteurs de Support (SVMs)

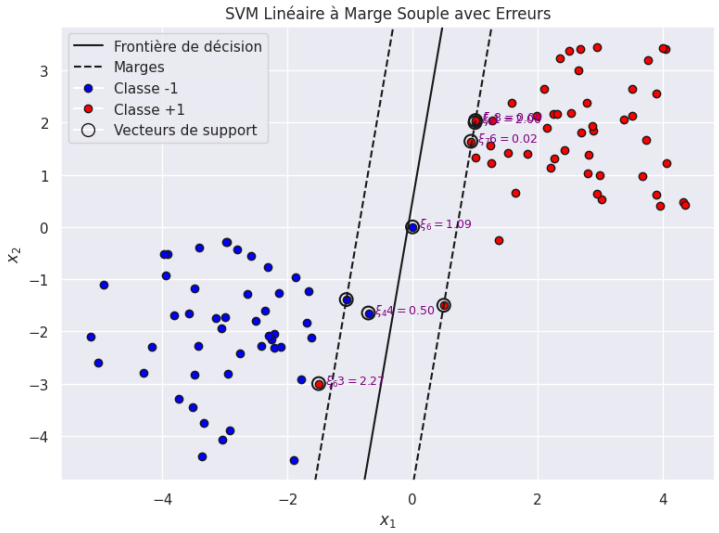
- Il devient alors nécessaire d'autoriser quelques violations de la marge, voire des erreurs de classification.
- C'est exactement ce que propose le **SVM à marge souple** (*Soft-Margin SVM*).



Machine à Vecteurs de Support (SVMs)

- Le principe de base reste inchangé : **Maximiser la marge.**
- Dans le cadre d'un **SVM à marge souple**, on permet à certains points de :
  - Être à l'intérieur de la marge.
  - Ou se trouver du mauvais côté de la frontière.
- Pour ce faire, on introduit des **variables de relâchement** $\xi_i \geq 0$ (*slack variables*) pour chaque point i. Ces variables mesurent l'écart par rapport à la contrainte de marge :

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i.$$

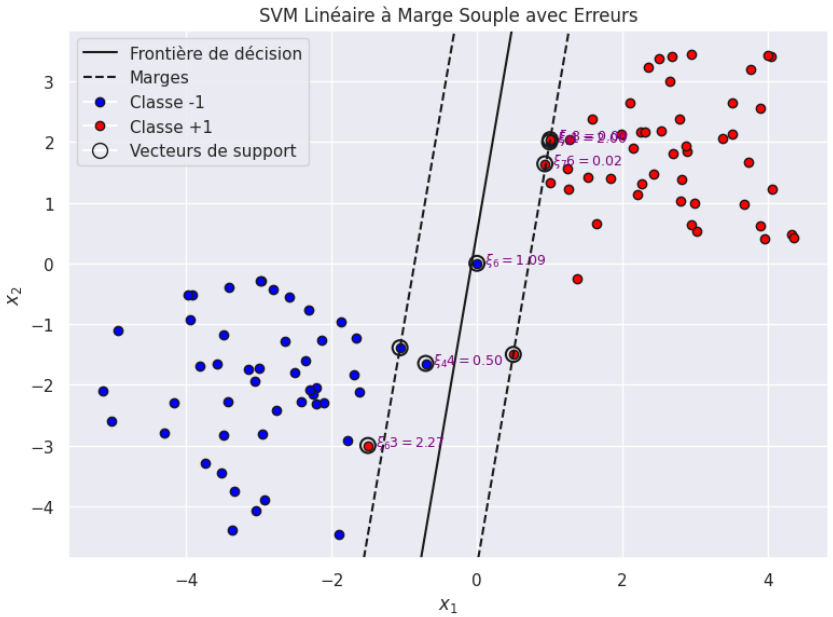


Machine à Vecteurs de Support (SVMs)

- On pénalise ces violations à l'aide d'un paramètre $C > 0$, qui contrôle le compromis entre la largeur de la marge et les erreurs de classification. L'objectif devient donc :

Objectif : Obtenir un compromis entre $\|\mathbf{w}\|^2$ (marge) et $\sum \xi_i$ (erreurs).

$$\equiv \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i.$$



Machine à Vecteurs de Support (SVMs)

- L'objectif est de maximiser la marge tout en tolérant certaines erreurs de classification.
- **Formulation du problème primal :**

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

Sous les contraintes :

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i.$$

Machine à Vecteurs de Support (SVMs)

- Nous introduisons les multiplicateurs de Lagrange $\alpha_i \geq 0$ pour chaque contrainte $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i$. Le Lagrangien du problème primal est donné par :

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 + \xi_i\right]$$

- Les résultats du calcul du gradient sont exactement les mêmes que ceux obtenus pour le SVM à marge rigide et les conditions du point selle sont exactement les mêmes, sauf pour la dérivée par rapport à ξ_i, qui donne $\alpha_i = C$.
- Calculons la dérivée de $\mathcal{L}$ par rapport à ξ_i :

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i.$$

- On obtient alors les conditions du point selle :

$$C - \alpha_i = 0 \quad \Rightarrow \quad \alpha_i = C.$$

Machine à Vecteurs de Support (SVMs)

- On obtient alors les conditions du point selle :

$$C - \alpha_i = 0 \quad \Rightarrow \quad \alpha_i = C.$$

- Cela signifie que le multiplicateur de Lagrange α_i prend la valeur C pour tous les points violant la contrainte de marge (c'est-à-dire ceux où $\xi_i > 0$).
- En substituant les résultats précédents dans l'expression Lagrangienne, on obtient la formulation du problème dual. Le problème dual est donc :

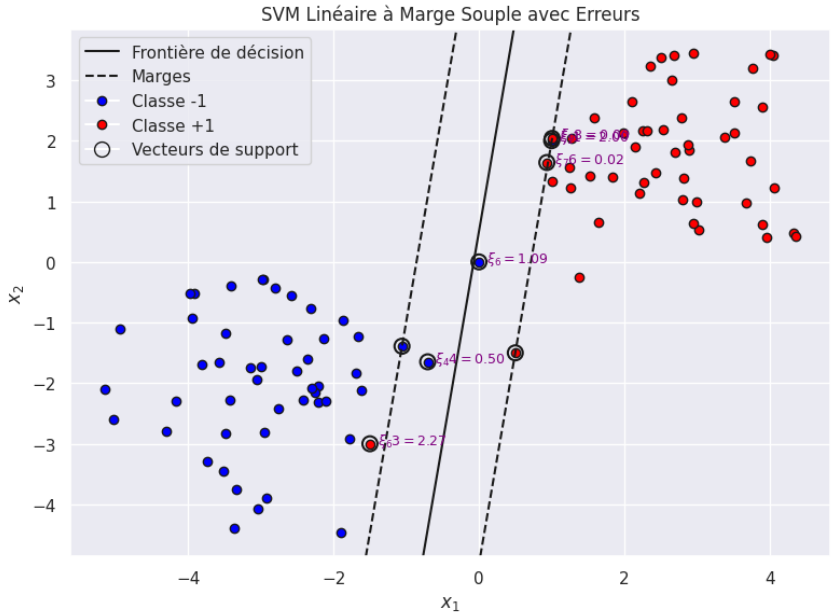
$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

sous les contraintes :

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

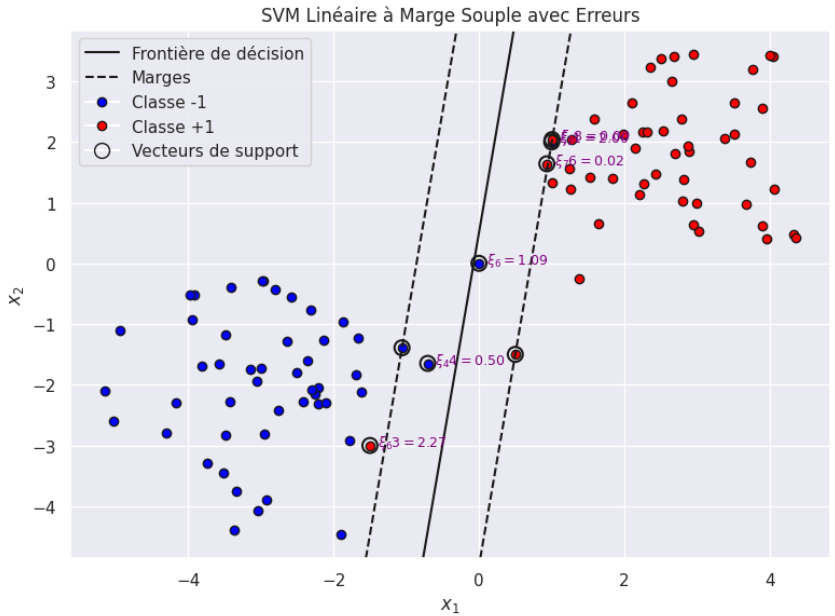
Machine à Vecteurs de Support (SVMs)

- Dans le problème dual, l'optimisation se fait par rapport aux multiplicateurs α_i, qui mesurent l'importance de chaque exemple dans la définition de l'hyperplan optimal.
- Ces multiplicateurs sont non nuls uniquement pour les vecteurs de support (les points les plus proches de la frontière). Ils jouent donc un rôle crucial dans le compromis entre la maximisation de la marge et la tolérance aux erreurs :



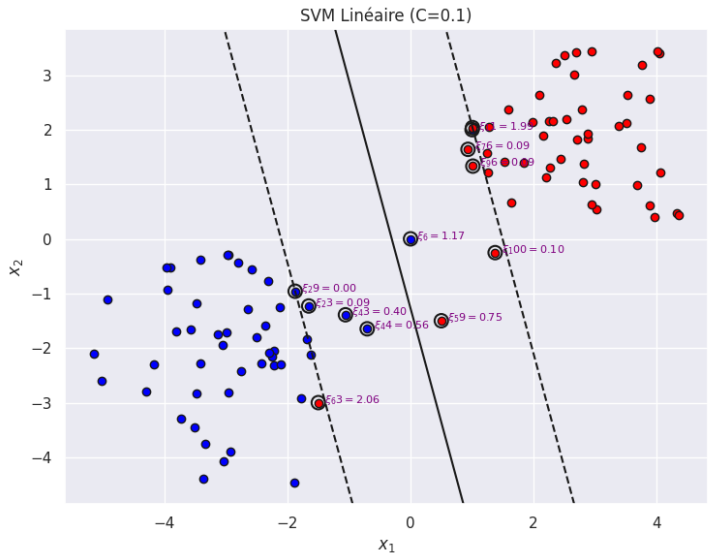
Machine à Vecteurs de Support (SVMs)

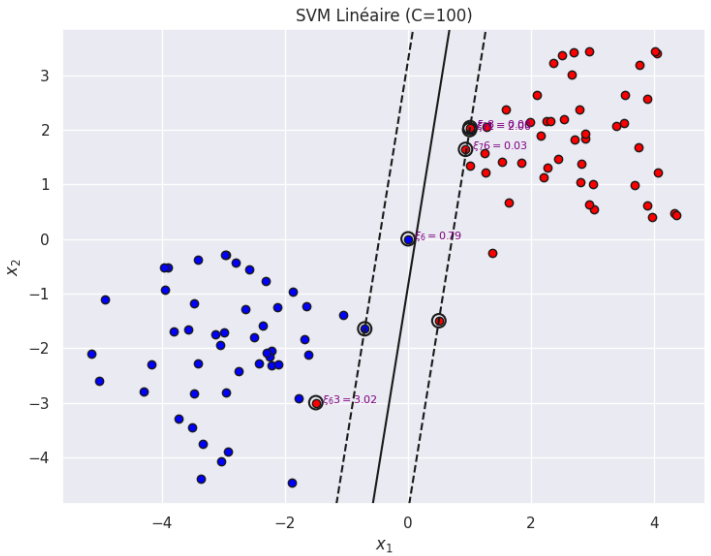
- Lorsque $\alpha_i > 0$, cela signifie que l'exemple $(\mathbf{x}_i, y_i)$ influence directement la position de l'hyperplan.
- Plus α_i est grand, plus l'exemple exerce une pression sur l'optimisation, ce qui réduit la marge pour permettre une meilleure classification de cet exemple.
- Si $\alpha_i = 0$, alors l'exemple $(\mathbf{x}_i, y_i)$ ne contribue pas à la définition de l'hyperplan et n'affecte pas la solution finale.



Machine à Vecteurs de Support (SVMs)

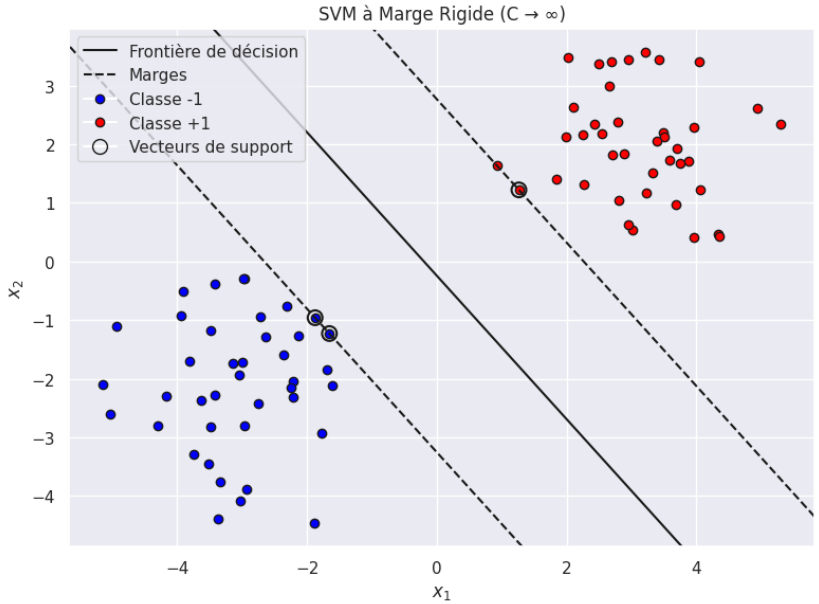
- La relation entre α_i et le paramètre C est fondamentale :
  - C contrôle l'équilibre entre la maximisation de la marge (réduire $\|\mathbf{w}\|^2$) et la pénalisation des erreurs de classification (réduire $\sum \xi_i$).
  - Lorsque C est grand, le modèle accorde une priorité plus grande à la réduction des erreurs de classification, ce qui entraîne des valeurs de α_i plus grandes pour les points mal classifiés. Cela peut réduire la marge en raison des violations des contraintes.
  - Lorsque C est petit, le modèle tolère davantage les erreurs de classification, ce qui permet d'augmenter la marge, mais au risque de classer certains points mal.





Machine à Vecteurs de Support (SVMs)

- Le **SVM à marge rigide** suppose que les données sont parfaitement séparables, sans aucune tolérance à l'erreur.
- Ce cas correspond à un **SVM à marge souple** avec un **paramètre** $C \to \infty$.
- Lorsque C est très grand (ex. pratique $C = 10^6$):
  - Le modèle ne tolère aucune erreur de classification.
  - Toute violation des contraintes de marge entraîne un coût énorme.
  - Le classifieur est donc contraint de séparer les données avec une marge maximale stricte.

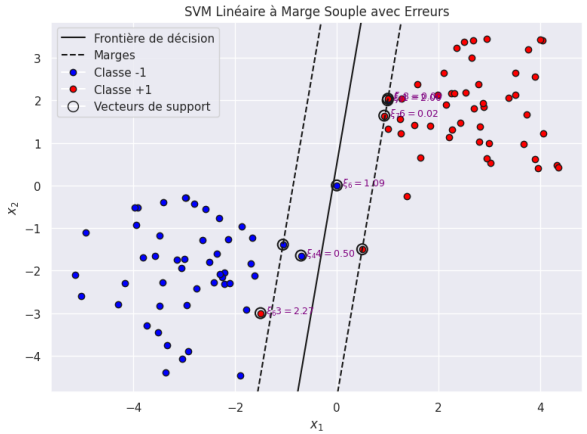


Machine à Vecteurs de Support (SVMs)

- La classification d'un nouvel exemple $\mathbf{x}_k$ se fait via la fonction de décision :

$$f(\mathbf{x_k}) = \mathbf{w}^\top \mathbf{x}_k + b = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x}_k + b$$

- En pratique, seuls les points pour lesquels $\alpha_i > 0$ interviennent dans cette somme :
  - Ce sont les **vecteurs de support**, les seuls exemples qui influencent l'hyperplan.
  - Cela rend le modèle **éparse** : la prédiction ne dépend que de quelques points-clés.
- La classification finale est donnée par le signe de $f(\mathbf{x_k})$.



Machine à Vecteurs de Support (SVMs) : Astuce du Noyau

- **Problème** : Même avec une marge souple, certaines classes ne sont **pas séparables linéairement**.
- **Solution** : Transformer les données dans un espace de plus grande dimension via une fonction non linéaire $\phi : \mathbb{R}^d \to \mathcal{H}$, dans lequel une séparation linéaire devient possible.
- **Astuce du Noyau (kernel trick)** : On ne calcule jamais explicitement $\phi(\mathbf{x})$, mais seulement des produits scalaires $\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$, remplacés par une fonction noyau $K(\mathbf{x}_i, \mathbf{x}_j)$.
- **Formulation duale généralisée** :

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \, K(\mathbf{x}_i, \mathbf{x}_j)$$

sous les contraintes :

$$0 \leq \alpha_i \leq C, \quad \sum_i \alpha_i y_i = 0$$

Machine à Vecteurs de Support (SVMs) : Astuce du Noyau

- **Limite du SVM linéaire** : Même avec une marge souple, certaines données ne sont toujours pas séparables linéairement.
- **Idée clé** : Transformer les données dans un espace de plus grande dimension via une application non linéaire :

$$\phi : \mathbb{R}^d \to \mathcal{H}, \quad \mathbf{x} \mapsto \phi(\mathbf{x}),$$

où une séparation linéaire devient possible.

- **Difficulté** : Calculer explicitement $\phi(\mathbf{x})$ peut être coûteux, voire impossible.
- **Astuce du noyau (kernel trick)** : Il n'est pas nécessaire de connaître ϕ, car la formulation duale du SVM repose uniquement sur des produits scalaires :

$$\text{Au lieu de} \quad \mathbf{x}_i^\top \mathbf{x}_j, \quad \text{on aura} \quad \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$
$$\text{qui sera remplacé par} \quad K(\mathbf{x}_i, \mathbf{x}_j)$$

Machine à Vecteurs de Support (SVMs) : Astuce du Noyau

- **Formulation duale généralisée** :

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

sous les contraintes :

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

- **Noyau linéaire** : $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$
- **Noyau polynôme** : $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + c)^d$
- **Noyau Gaussien (RBF)** :

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

- **Remarques** :
  - γ contrôle la largeur du noyau RBF : un petit γ crée des régions larges, un grand γ donne une frontière plus serrée.
  - Le noyau RBF permet de séparer des classes très tordues (ex: cercles imbriqués).

Machine à Vecteurs de Support (SVMs) : Astuce du Noyau

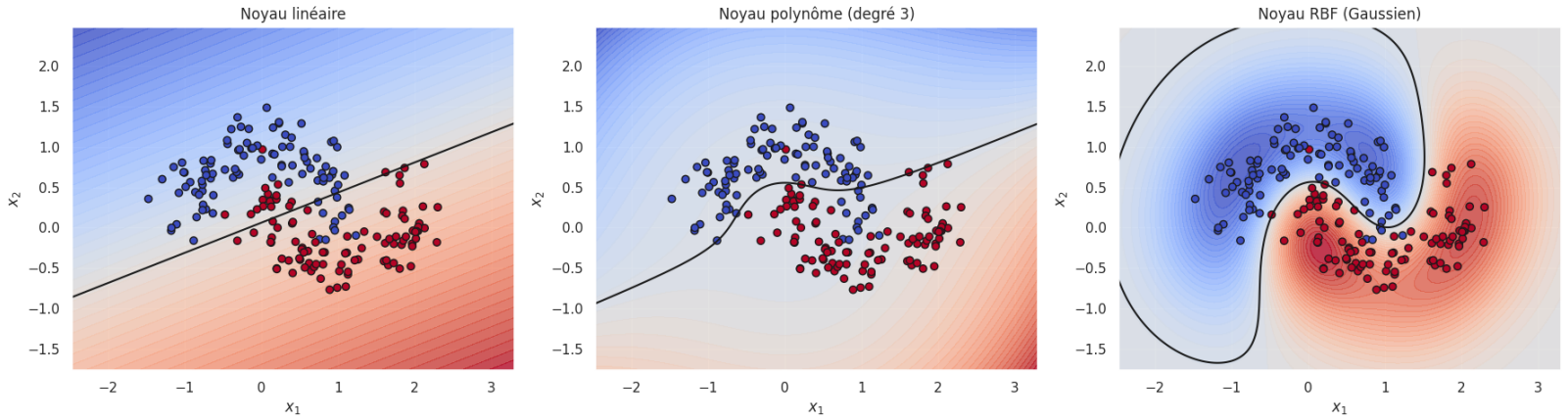
- **Exemples de noyaux** :
  - **Noyau linéaire** : $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$
  - **Noyau polynôme** : $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + c)^d$
  - **Noyau Gaussien (RBF)** :

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

- **Remarques importantes sur le noyau RBF** :
  - Il correspond à un produit scalaire dans un espace de Hilbert **de dimension infinie**, obtenu par une transformation non linéaire ϕ (inconnue en pratique).
  - L'astuce du noyau permet **d'éviter d'expliciter** $\phi(\mathbf{x})$: on ne calcule que les produits scalaires $\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$ via $K(\mathbf{x}_i, \mathbf{x}_j)$.
  - Ainsi, **on peut construire des frontières hautement non linéaires sans jamais quitter l'espace d'origine.**
  - Le paramètre γ contrôle la complexité du modèle : plus γ est grand, plus la frontière de décision est fine et localisée.

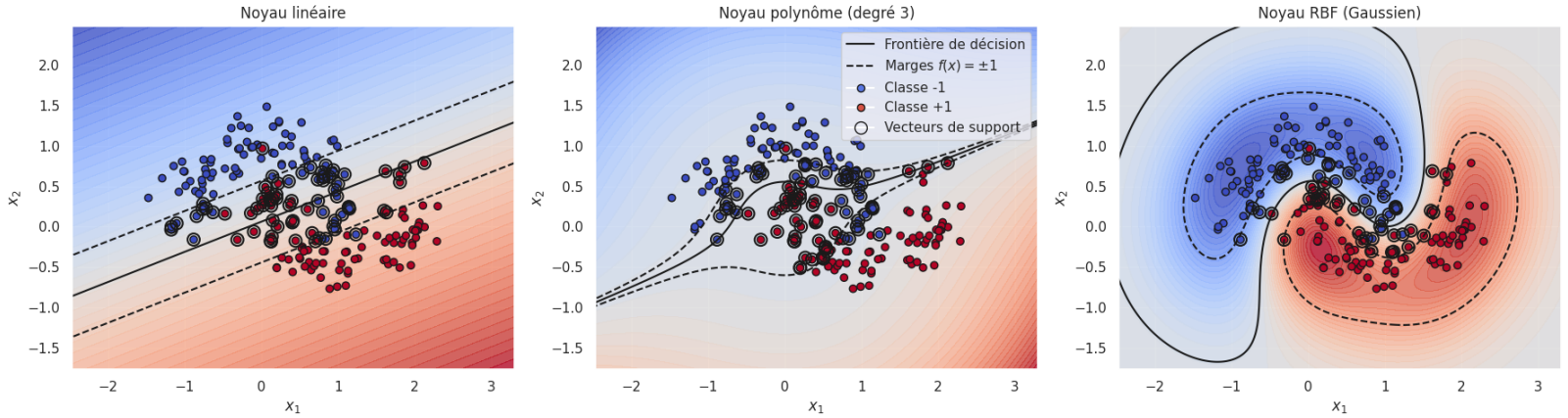
Machine à Vecteurs de Support (SVMs) : Astuce du Noyau

- **Noyau linéaire** :
  - Produit scalaire standard : $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$
  - La frontière est une droite : adaptée uniquement aux cas linéairement séparables.
  - Limité quand les données sont non linéaires.
- **Noyau polynôme (degré 3)** :
  - $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + c)^d$
  - Crée des frontières courbes et plus flexibles.
  - Peut mieux s'adapter aux données non linéaires que le noyau linéaire.
- **Noyau Gaussien (RBF)** :
  - $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$
  - Permet de modéliser des frontières non linéaires et tordues.



Machine à Vecteurs de Support (SVMs) : Astuce du Noyau

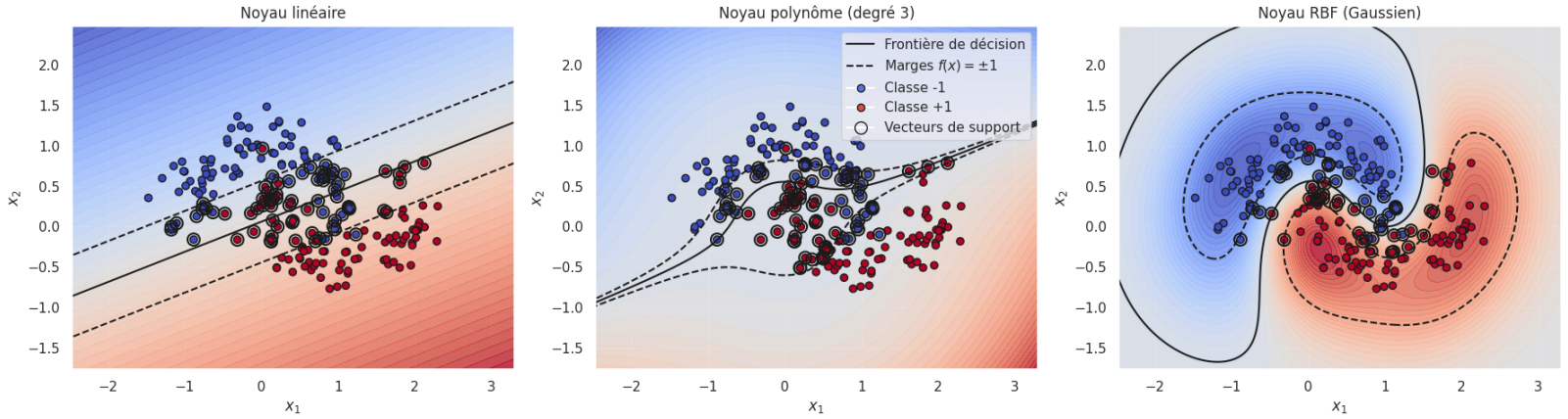
- Les **vecteurs de support** (cerclés en noir) sont les points qui influencent directement l'hyperplan. Ils peuvent être :
  - sur la marge : $y_i f(x_i) = 1$
  - dans la marge : $0 < y_i f(x_i) < 1$
  - **mal classés** : $y_i f(x_i) < 0$
- Même si $y_i f(x_i) < -1$, un point peut être vecteur de support si $\alpha_i > 0$, car le SVM cherche un compromis entre **maximisation de la marge** et **minimisation des erreurs**. Ce point est fortement pénalisé, mais il contribue à la solution optimale.



Machine à Vecteurs de Support (SVMs) : Astuce du Noyau

Situation	$y_i f(x_i)$	ξ_i	α_i
Bien classé, hors marge	> 1	$= 0$	$= 0$
Sur la marge	$= 1$	$= 0$	$0 < \alpha_i < C$
À l'intérieur de la marge	$0 < y_i f(x_i) < 1$	$0 < \xi_i < 1$	$\alpha_i = C$
Mal classé	< 0	$\xi_i > 1$	$\alpha_i = C$

- Seuls les exemples avec $\alpha_i > 0$ sont des **vecteurs de support**.
- Les points mal classés $(y_i f(x_i) < 0)$ ont un coût élevé dans la fonction objectif via ξ_i, mais ils participent à la définition de la frontière.

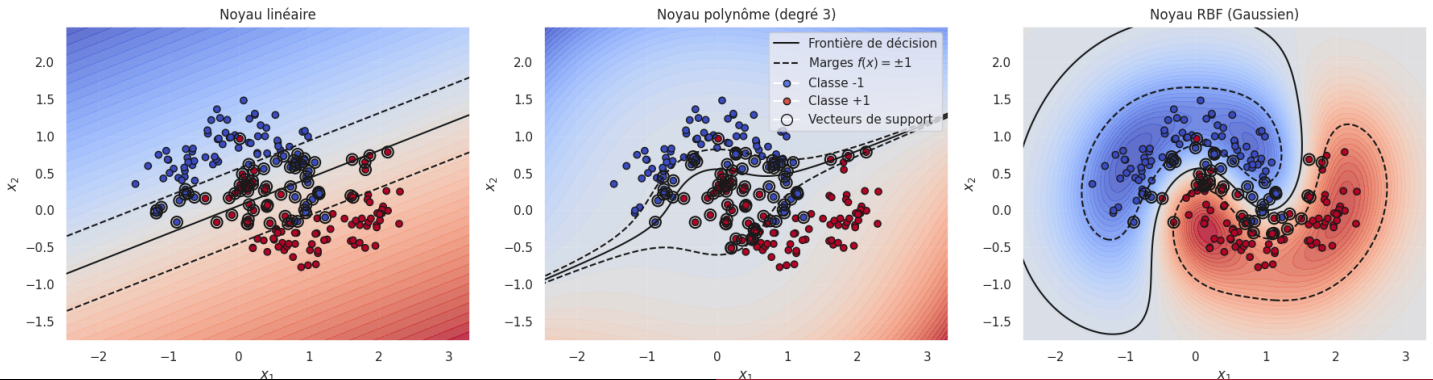


Machine à Vecteurs de Support (SVMs) : Astuce du Noyau

- La classification pour un nouvel exemple $\mathbf{x}_k$ est donnée par la fonction suivante :

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_k) + b.$$

- $K(\mathbf{x}_i, \mathbf{x}_k)$ est le noyau qui mesure la similarité entre l'exemple d'entraînement $\mathbf{x}_i$ et le nouvel exemple $\mathbf{x}_k$.
- L'idée clé ici est que la prédiction repose uniquement sur les points les plus proches de la frontière de décision, c'est-à-dire les vecteurs de support, qui ont $\alpha_i > 0$.
- Si $f(\mathbf{x}_k) > 0$, alors $\mathbf{x}_k$ est classé dans la classe $+1$.
- Si $f(\mathbf{x}_k) < 0$, alors $\mathbf{x}_k$ est classé dans la classe -1.



Machine à Vecteurs de Support (SVMs) : SMO

- SMO (Sequential Minimal Optimization) est une méthode efficace pour résoudre le problème d'optimisation dans l'entraînement des SVM à marge souple.
- L'objectif de SMO est de maximiser la marge tout en minimisant les erreurs de classification, en mettant à jour progressivement les coefficients α_i des vecteurs de support.

Machine à Vecteurs de Support (SVMs) : SMO

- Nous avons vu que le problème d'optimisation d'un SVM est formulé comme suit :

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

Sous les contraintes :

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

où α_i sont les multiplicateurs de Lagrange, y_i sont les étiquettes, C est un paramètre de régularisation, et $K(x_i, x_j)$ est le noyau (par exemple, un noyau linéaire ou RBF).

- **Principe de l'algorithme SMO :** L'algorithme SMO décompose le problème complexe en sous-problèmes plus simples en mettant à jour deux multiplicateurs de Lagrange α_i et α_j à la fois, au lieu de tous les coefficients à la fois.

Machine à Vecteurs de Support (SVMs) : SMO

- L'algorithme sélectionne deux indices i et j parmi les n points de données.
- Ces indices sont choisis pour optimiser l'objectif de la fonction.
- Typiquement, i est choisi parmi les points avec la plus grande erreur (points mal classifiés ou proches de la frontière de décision).
- L'objectif est de mettre à jour les coefficients α_i et α_j pour réduire l'erreur du modèle.
- Ces coefficients sont mis à jour en fonction des gradients de la fonction objectif par rapport à α_i et α_j.
- Les mises à jour respectent les contraintes $0 \leq \alpha_i \leq C$ et $\sum_{i=1}^{n} \alpha_i y_i = 0$.

Machine à Vecteurs de Support (SVMs) : SMO

- **Calcul des dérivées :** Les dérivées par rapport aux α_i et α_j sont calculées pour déterminer l'importance des points mal classifiés. Ces dérivées proviennent de la fonction de décision, qui évalue la position d'un point par rapport à l'hyperplan de séparation.

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = y_i \left(f(\mathbf{x}_i) - y_i \right)$$

où $f(\mathbf{x}_i)$ est la fonction de décision du modèle. La fonction de décision $f(\mathbf{x}_i)$ est définie comme suit :

$$f(\mathbf{x}_i) = \sum_{k=1}^{n} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_i) + b$$

Ici, $K(\mathbf{x}_k, \mathbf{x}_i)$ est le noyau (par exemple, linéaire ou RBF) qui mesure la similitude entre les points $\mathbf{x}_k$ et $\mathbf{x}_i$, α_k est le coefficient associé à chaque vecteur de support $\mathbf{x}_k$, y_k est l'étiquette de classe, et b est le biais.

Machine à Vecteurs de Support (SVMs) : SMO

- **Contrôles de bornes :** Les valeurs de α_i et α_j sont ajustées pour respecter les bornes imposées par le problème d'optimisation. Les bornes sont calculées en fonction des valeurs de α_i et α_j, et elles doivent garantir que ces coefficients restent dans les limites imposées par le problème d'optimisation.
- Les bornes L et H sont définies comme suit :

$$L = \max(0, \alpha_j - \alpha_i) \quad \text{et} \quad H = \min(C, C + \alpha_j - \alpha_i)$$

Ces bornes garantissent que les coefficients α_i et α_j restent dans la plage $[0, C]$, **où C est le paramètre de régularisation qui contrôle la tolérance aux erreurs.**

- L'algorithme veille également à ce que la condition suivante soit satisfaite après chaque mise à jour des coefficients :

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

Cette condition est une contrainte nécessaire pour que l'optimisation des coefficients soit valide.

Machine à Vecteurs de Support (SVMs) : SMO

- **Mise à jour du biais b :** Après la mise à jour des α_i et α_j, le biais b est recalculé pour maintenir l'équilibre de la fonction de décision. Le biais b est calculé en prenant la moyenne des deux biais intermédiaires b_1 et b_2, qui sont obtenus à partir des erreurs des points i et j :

$$b = \frac{b_1 + b_2}{2}$$

où les expressions de b_1 et b_2 sont données par :

$$\begin{aligned} b_1 = b - E_i - y_i \left(\alpha_i - \alpha_i^{\text{old}}\right) K(\mathbf{x}_i, \mathbf{x}_i) \\ - y_j \left(\alpha_j - \alpha_j^{\text{old}}\right) K(\mathbf{x}_i, \mathbf{x}_j). \end{aligned}$$

$$\begin{aligned} b_2 = b - E_j - y_i \left(\alpha_i - \alpha_i^{\text{old}}\right) K(\mathbf{x}_i, \mathbf{x}_j) \\ - y_j \left(\alpha_j - \alpha_j^{\text{old}}\right) K(\mathbf{x}_j, \mathbf{x}_j). \end{aligned}$$

Machine à Vecteurs de Support (SVMs) : SMO

- **Expressions des erreurs E_i et E_j :**
  - E_i est l'erreur de prédiction pour le point i :

$$E_i = f(\mathbf{x}_i) - y_i$$

où $f(\mathbf{x}_i)$ est la fonction de décision du modèle, définie par :

$$f(\mathbf{x}_i) = \sum_{k=1}^{n} \alpha_k y_k K(\mathbf{x}_k, \mathbf{x}_i) + b$$

  - E_j est l'erreur de prédiction pour le point j :

$$E_j = f(\mathbf{x}_j) - y_j$$

- **Critères d'arrêt :** L'algorithme continue à effectuer des mises à jour des coefficients α_i jusqu'à ce que l'un des critères suivants soit atteint :
  - Un nombre maximal de passes sur l'ensemble des données est atteint.
  - Aucun changement significatif n'est effectué sur les coefficients α_i, ce qui signifie que les valeurs de α ont convergé.

Machine à Vecteurs de Support (SVMs) : SMO

- L'algorithme est très efficace car il ne nécessite pas de calculer la matrice Hessienne complète, ce qui accélère les calculs par rapport à d'autres méthodes d'optimisation.
- Il est adapté aux grandes bases de données car il effectue des mises à jour locales et itératives.
- **Résumé :** SMO résout le problème d'optimisation des SVM en sélectionnant deux multiplicateurs à la fois et en les mettant à jour de manière itérative tout en respectant les contraintes. Cela le rend particulièrement adapté aux SVM à marge souple, notamment lorsqu'on utilise des noyaux comme le noyau RBF.

Machine à Vecteurs de Support (SVMs) : Exercice SMO

- L'objectif de cet exercice est d'implémenter l'algorithme SMO (Sequential Minimal Optimization) pour entraîner un SVM à marge souple avec noyau RBF.
- L'algorithme SMO permet de résoudre le problème d'optimisation des SVM en mettant à jour deux multiplicateurs de Lagrange à la fois, au lieu de tous les coefficients simultanément, et de manière itérative.
- Nous allons d'abord implémenter l'algorithme SMO pour un noyau linéaire, puis le généraliser pour d'autres noyaux (polynôme, RBF).
- **Exercice** : Implémentez SMO et testez-le sur un jeu de données non linéaires (par exemple, *make moons* de scikit-learn).

Initialisation et Importation des Bibliothèques

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
import seaborn as sns
from matplotlib.lines import Line2D

# Style graphique
sns.set_theme()
```

Fonction de Prédiction - SMO

```python
# Fonction de prédiction
def smo(X, y, C=1.0, tol=1e-3, max_passes=5, kernel="linear", gamma=1.0, degree=3):
    m, n = X.shape
    alpha = np.zeros(m)  # Coefficients alpha
    b = 0  # Le biais
    passes = 0

    # Calcul du produit scalaire X^T X (pour les noyaux linéaires)
    if kernel == "linear":
        K = np.dot(X, X.T)
    elif kernel == "poly":
        K = (np.dot(X, X.T) + 1)**degree
    elif kernel == "rbf":
        K = np.exp(-gamma * np.linalg.norm(X[:, np.newaxis] - X, axis=2)**2)
```

Boucle de Mise à Jour des Coefficients et Biais

```
while passes < max_passes:
    num_changed_alphas = 0

    for i in range(m):
        # Calcul de la fonction de décision f(x_i) pour un exemple i
        fxi = np.sum(alpha * y * K[i]) + b
        Ei = fxi - y[i]  # Erreur de prédiction

        if (y[i] * Ei < -tol and alpha[i] < C) or (y[i] * Ei > tol and alpha[i] > 0):
            # Sélectionner un autre alpha j
            j = np.random.choice([x for x in range(m) if x != i])

            fxj = np.sum(alpha * y * K[j]) + b
            Ej = fxj - y[j]  # Erreur de prédiction pour j

            # Sauvegarder les valeurs précédentes de alpha[i] et alpha[j]
            alpha_i_old, alpha_j_old = alpha[i], alpha[j]

            # Calcul de la borne (le taux de mise à jour de alpha)
            if y[i] != y[j]:
                L = max(0, alpha[j] - alpha[i])
                H = min(C, C + alpha[j] - alpha[i])
            else:
                L = max(0, alpha[i] + alpha[j] - C)
                H = min(C, alpha[i] + alpha[j])

            if L == H:
                continue
```

Calcul de la Dérivée et Mise à Jour de alpha

```
            # Calcul de la dérivée
            eta = 2 * K[i, j] - K[i, i] - K[j, j]

            if eta >= 0:
                continue

            # Mise à jour de alpha_j
            alpha[j] -= (y[j] * (Ei - Ej)) / eta
            # Clamping alpha_j entre les bornes L et H
            alpha[j] = np.clip(alpha[j], L, H)

            if abs(alpha[j] - alpha_j_old) < tol:
                continue

            # Mise à jour de alpha_i
            alpha[i] += y[i] * y[j] * (alpha_j_old - alpha[j])

            # Mise à jour du biais b
            b1 = b - Ei - y[i] * (alpha[i] - alpha_i_old) * K[i, i] - y[j] * (alpha[j] - alpha_j_old) * K[i, j]
            b2 = b - Ej - y[i] * (alpha[i] - alpha_i_old) * K[i, j] - y[j] * (alpha[j] - alpha_j_old) * K[j, j]
            b = (b1 + b2) / 2

            num_changed_alphas += 1

    if num_changed_alphas == 0:
        passes += 1
    else:
        passes = 0

return alpha, b
```

Entraînement du modèle avec SMO

```
# Générer les données de type "moon"
X, y = make_moons(n_samples=200, noise=0.2, random_state=42)

# Convertir les étiquettes en {-1, 1}
y = 2 * y - 1

# Paramètres pour l'implémentation SMO (réglages de base)
C = 1.0  # Pénalité
tol = 1e-3  # Tolérance pour les erreurs
max_passes = 5  # Nombre de passes maximales

# Entraînement du modèle avec SMO
alpha, b = smo(X, y, C=1.0, tol=1e-3, max_passes=5, kernel="rbf")
```

Visualisation des Résultats

```
# Calcul de la fonction de décision pour chaque point du maillage
xx, yy = np.meshgrid(np.linspace(X[:, 0].min() - 1, X[:, 0].max() + 1, 500),
                     np.linspace(X[:, 1].min() - 1, X[:, 1].max() + 1, 500))
# Utilisation du noyau RBF pour calculer les valeurs de décision
gamma = 1.0
Z = []
for x_new in np.c_[xx.ravel(), yy.ravel()]:
    k_vals = np.exp(-gamma * np.linalg.norm(X - x_new, axis=1)**2)
    f_x = np.sum(alpha * y * k_vals) + b
    Z.append(f_x)
Z = np.array(Z).reshape(xx.shape)
# Tracé
plt.figure(figsize=(8, 6))
# Visualisation des contours des marges et de la frontière de décision
plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), Z.max(), 50), cmap=plt.cm.coolwarm, alpha=0.8)
plt.contour(xx, yy, Z, levels=[0], colors='k', linewidths=1.5)
plt.contour(xx, yy, Z, levels=[-1, 1], colors='k', linestyles='--', linewidths=1.5)
# Visualisation des données
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm, edgecolors='k')
# Visualisation des vecteurs de support
support_vectors = X[np.where(alpha > 0)[0]]
plt.scatter(support_vectors[:, 0], support_vectors[:, 1], facecolors='none', edgecolors='k', s=100, linewidths=1.5, label="Vect
# Légende manuelle
legend_elements = [
    Line2D([0], [0], color='k', linestyle='-', label='Frontière de décision'),
    Line2D([0], [0], color='k', linestyle='--', label='Marges'),
    Line2D([0], [0], marker='o', color='w', markerfacecolor='blue',
           markeredgecolor='k', label='Classe -1'),
    Line2D([0], [0], marker='o', color='w', markerfacecolor='red',
           markeredgecolor='k', label='Classe +1'),
    Line2D([0], [0], marker='o', color='w', markerfacecolor='none',
           markeredgecolor='k', markersize=10, label='Vecteurs de support')
]
```

Visualisation des Résultats

```
plt.legend(handles=legend_elements)
plt.title("SVM à Marge Souple avec Noyau RBF")
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")
plt.grid(True)
plt.tight_layout()
plt.show()
```

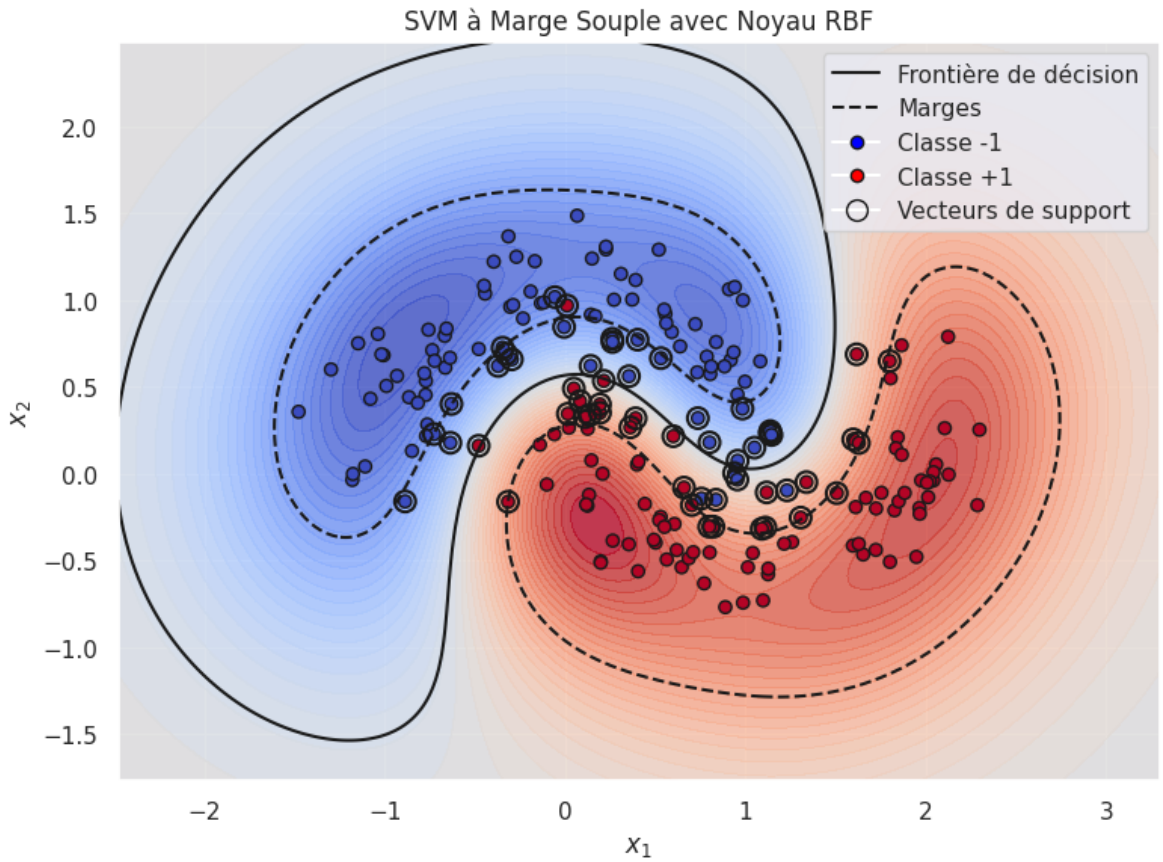


Table des Matières

1 Modèles Linéaires Généralisés

2 Regression Logistique

3 Analyse Linéaire Discriminante

4 Annexe

Annexe

Dérivation des Moments de la Famille Exponentielle

Dérivation des Moments de la Famille Exponentielle

- Soit la fonction de densité de la famille exponenitelle écrite sous la forme :

$$f_Y(y;\theta,\phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right).$$

On souhaite démontrer tout d'abord que $\mathbb{E}[Y] = b'(\theta)$.

- **Étape 1 : Calcul de la Log-Vraisemblance**

$$\ell(\theta,\phi \mid y) = \log\left[f_Y(y;\theta,\phi)\right] = \frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi).$$

- **Étape 2 : Calcul de la fonction Score $\frac{\partial \ell}{\partial \theta}$**

$$\frac{\partial \ell}{\partial \theta} = \frac{y - b'(\theta)}{a(\phi)}.$$

- **Étape 3 — Calcul de l'Espérance du Score $\mathbb{E}\left[\frac{\partial \ell}{\partial \theta}\right]$**

$$\mathbb{E}\left[\frac{\partial \ell}{\partial \theta}\right] = \mathbb{E}\left[\frac{y - b'(\theta)}{a(\phi)}\right] = \frac{\mathbb{E}[Y] - b'(\theta)}{a(\phi)}.$$

Dérivation des Moments de la Famille Exponentielle

- L'espérance du Score est donnée par :

$$\begin{aligned}
\mathbb{E}\left[\frac{\partial \ell}{\partial \theta}\right] &= \mathbb{E}\left[\frac{\log\left[f_Y(y;\theta,\phi)\right]}{\partial \theta}\right] \\
&= \int \frac{1}{f_Y(y;\theta,\phi)} \cdot \frac{\partial f_Y(y;\theta,\phi)}{\partial \theta} \cdot f_Y(y;\theta,\phi)\, dy \\
&= \int \frac{f_Y(y;\theta,\phi)}{f_Y(y;\theta,\phi)} \cdot \frac{\partial f_Y(y;\theta,\phi)}{\partial \theta} \cdot dy \\
&= \int \frac{\partial f_Y(y;\theta,\phi)}{\partial \theta}\, dy \\
&= \frac{\partial}{\partial \theta} \int f(y;\theta,\phi) dy \\
&= \frac{\partial}{\partial \theta}(1) = 0
\end{aligned}$$

- $\mathbb{E}\left[\frac{\partial \ell}{\partial \theta}\right] = 0 \Rightarrow \frac{\mathbb{E}[Y] - b'(\theta)}{a(\phi)} = 0 \Rightarrow \mathbb{E}[Y] = b'(\theta)$ (CQFD).

Dérivation des Moments de la Famille Exponentielle

- On souhaite maintenant démontrer que $\text{Var}(Y) = b''(\theta) \cdot a(\phi)$ en exploitant la dérivée seconde de la log-vraisemblance.
- Fonction log-vraisemblance :

$$\ell(\theta, \phi \mid y) = \frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)$$

- Fonction score :

$$\frac{\partial \ell}{\partial \theta} = \frac{y - b'(\theta)}{a(\phi)}$$

- Dérivée seconde :

$$\frac{\partial^2 \ell}{\partial \theta^2} = \frac{-b''(\theta)}{a(\phi)}, \quad \textbf{Résultat (I).}$$

- On a besoin de démontrer l'identité de **l'Information de Fisher** :

$$\mathbb{E}\left[\left(\frac{\partial \ell}{\partial \theta}\right)^2\right] = -\mathbb{E}\left[\frac{\partial^2 \ell}{\partial \theta^2}\right]$$

Dérivation des Moments de la Famille Exponentielle

- On rappelle que la fonction log-vraisemblance est :

$$\ell(\theta, \phi \mid y) = \log\left[f_Y(y; \theta, \phi)\right]$$

- On cherche à calculer :

$$\mathbb{E}\left[\frac{\partial^2 \ell}{\partial \theta^2}\right] = \mathbb{E}\left[\frac{\partial^2}{\partial \theta^2} \log\left[f_Y(y; \theta, \phi)\right]\right]$$

- On applique la dérivée du logarithme d'une fonction :

$$\frac{\partial}{\partial \theta} \log(f_Y) = \frac{1}{f_Y} \cdot \frac{\partial f_Y}{\partial \theta}$$

- Donc, la dérivée seconde s'écrit :

$$\mathbb{E}\left[\frac{\partial^2 \ell}{\partial \theta^2}\right] = \frac{\partial^2}{\partial \theta^2} \log(f_Y) = \frac{\partial}{\partial \theta}\left(\frac{1}{f_Y} \cdot \frac{\partial f_Y}{\partial \theta}\right)$$

Dérivation des Moments de la Famille Exponentielle

- On applique la règle du quotient et de la chaîne :

$$\begin{aligned}\mathbb{E}\left[\frac{\partial^2 \ell}{\partial \theta^2}\right] &= \frac{\partial}{\partial \theta}\left(\frac{1}{f_Y}\cdot\frac{\partial f_Y}{\partial \theta}\right) = \frac{\partial}{\partial \theta}\left(f_Y^{-1}\cdot\frac{\partial f_Y}{\partial \theta}\right)\\ &= \left(-\frac{1}{f_Y^2}\cdot\frac{\partial f_Y}{\partial \theta}\right)\cdot\frac{\partial f_Y}{\partial \theta} + \frac{1}{f_Y}\cdot\frac{\partial^2 f_Y}{\partial \theta^2}\\ &= -\frac{1}{f_Y^2}\left(\frac{\partial f_Y}{\partial \theta}\right)^2 + \frac{1}{f_Y}\cdot\frac{\partial^2 f_Y}{\partial \theta^2}\end{aligned}$$

- Ainsi, on a :

$$\begin{aligned}\mathbb{E}\left[\frac{\partial^2 \ell}{\partial \theta^2}\right] &= \mathbb{E}\left[-\frac{1}{f_Y^2}\left(\frac{\partial f_Y}{\partial \theta}\right)^2 + \frac{1}{f_Y}\cdot\frac{\partial^2 f_Y}{\partial \theta^2}\right]\\ \Rightarrow \mathbb{E}\left[\frac{\partial^2 \ell}{\partial \theta^2}\right] &= -\mathbb{E}\left[\left(\frac{1}{f_Y}\cdot\frac{\partial f_Y}{\partial \theta}\right)^2\right] + \mathbb{E}\left[\frac{1}{f_Y}\cdot\frac{\partial^2 f_Y}{\partial \theta^2}\right]\\ &= -\mathbb{E}\left[\left(\frac{\partial \ell}{\partial \theta}\right)^2\right] + \mathbb{E}\left[\frac{1}{f_Y}\cdot\frac{\partial^2 f_Y}{\partial \theta^2}\right]\end{aligned}$$

Dérivation des Moments de la Famille Exponentielle

- On remarque que :

$$\begin{aligned}\mathbb{E}\left[\frac{1}{f_Y}\cdot\frac{\partial^2 f_Y}{\partial\theta^2}\right] &= \int \frac{1}{f_Y}\cdot\frac{\partial^2 f_Y}{\partial\theta^2}\cdot f_Y\,dy \\ &= \int \frac{\partial^2 f_Y}{\partial\theta^2}\,dy \\ &= \frac{\partial^2}{\partial\theta^2}\int f_Y(y;\theta,\phi)\,dy \\ &= \frac{\partial^2}{\partial\theta^2}(1) = 0\end{aligned}$$

- Donc :

$$\mathbb{E}\left[\frac{1}{f_Y(y;\theta,\phi)}\cdot\frac{\partial^2 f_Y(y;\theta,\phi)}{\partial\theta^2}\right] = 0$$

- Il en résulte que :

$$\mathbb{E}\left[\frac{\partial^2\ell}{\partial\theta^2}\right] = -\mathbb{E}\left[\left(\frac{\partial\ell}{\partial\theta}\right)^2\right], \quad \textbf{Résultat (II).}$$

Dérivation des Moments de la Famille Exponentielle

- On a :

$$\frac{\partial \ell}{\partial \theta} = \frac{y - b'(\theta)}{a(\phi)} \quad \Rightarrow \quad \left(\frac{\partial \ell}{\partial \theta}\right)^2 = \frac{(y - b'(\theta))^2}{a(\phi)^2}$$

- Alors :

$$\mathbb{E}\left[\left(\frac{\partial \ell}{\partial \theta}\right)^2\right] = \frac{\mathbb{E}\left[(Y - \mu)^2\right]}{a(\phi)^2} = \frac{\mathsf{Var}(Y)}{a(\phi)^2}$$

- D'autre part, d'après **Résultat (I)** :

$$-\mathbb{E}\left[\frac{\partial^2 \ell}{\partial \theta^2}\right] = \frac{b''(\theta)}{a(\phi)}.$$

- En égalant les deux expressions dans **Résultat (I)** et **Résultat (II)**:

$$\frac{b''(\theta)}{a(\phi)} = \frac{\mathsf{Var}(Y)}{a(\phi)^2} \Rightarrow \mathsf{Var}(Y) = b''(\theta) \cdot a(\phi), \quad \text{(CQFD)}.$$

Fonction de Score et Information de Fisher

Fonction de Score et Information de Fisher

- Toute variable aléatoire Y dont la loi appartient à la famille exponentielle a une densité de probabilité de la forme :

$$f_Y(y;\theta,\phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right)$$

- Dans ce cadre, on peut analyser :
  - **la fonction de score**, qui mesure la sensibilité de la log-vraisemblance par rapport à θ,
  - **l'information observée**, qui donne la courbure locale de la log-vraisemblance,
  - **l'information de Fisher**, qui mesure la précision espérée de l'estimateur.

Fonction de Score et Information de Fisher

- **Définition formelle :** Soit $Y \sim f_Y(y;\theta)$, une variable aléatoire dont la loi dépend d'un paramètre $\theta \in \mathbb{R}$. La log-vraisemblance est donnée par:

$$\boxed{\ell(\theta; y) = \log\left[f_Y(y;\theta)\right]}$$

La **fonction de score** $U(\theta)$ est la dérivée de la log-vraisemblance par rapport à θ :

$$\boxed{U(\theta) := \frac{\partial}{\partial\theta}\ell(\theta; y) = \frac{\partial}{\partial\theta}\log\left[f_Y(y;\theta)\right]}$$

- $U(\theta)$ est une variable aléatoire (car dépend de Y) :
  - Elle indique la direction dans laquelle modifier θ pour augmenter la vraisemblance.

Fonction de Score et Information de Fisher

- $U(\theta) = \frac{\partial \ell}{\partial \theta}$ agit comme une **boussole** qui guide l'optimisation de θ.
- **Intuition** : si on observe une donnée y, on se demande *Pour quelle valeur de θ cette donnée est-elle la plus probable ?*
  - Si $\frac{\partial \ell}{\partial \theta} > 0$: on peut augmenter θ pour améliorer la vraisemblance.
  - Si $\frac{\partial \ell}{\partial \theta} < 0$: on peut diminuer θ.
  - Si $\frac{\partial \ell}{\partial \theta} = 0$: on est sur un **point critique** (potentiellement maximum).
- **Résultat fondamental :** On a démontré dans **le calcul de la moyenne pour une densité faisant partie de la famille exponentielle que**

$$\mathbb{E}[U(\theta)] = \mathbb{E}\left[\frac{\partial \ell}{\partial \theta}\right] = 0.$$

- **Ceci est vrai pour n'importe quelle densité $f_Y(y;\theta)$:**
  - Le maximum de vraisemblance est atteint lorsque la dérivée de la log-vraisemblance est nulle en moyenne.

Fonction de Score et Information de Fisher

- **Définition** : **L'information observée** est l'opposée de la dérivée seconde de la log-vraisemblance :

$$\boxed{J(\theta) := -\frac{\partial^2 \ell(\theta; y)}{\partial \theta^2}}$$

- C'est une mesure locale de la courbure de la log-vraisemblance autour de θ.
- Si la log-vraisemblance est très courbée (pic étroit), alors l'information est grande en quantité et l'estimation est alors plus précise.
- Si elle est plate, alors l'estimation est plus incertaine.

Fonction de Score et Information de Fisher

- **L'information de Fisher** est l'espérance de l'information observée. Elle quantifie cette concavité moyenne :

$$\boxed{I(\theta) := \mathbb{E}_\theta \left[J(\theta) \right] = -\mathbb{E}_\theta \left[\frac{\partial^2 \ell(\theta; Y)}{\partial \theta^2} \right]}$$

- Lorsque l'on peut échanger dérivation et intégration, on a aussi :

$$\boxed{I(\theta) = \mathbb{E}_\theta \left[\left(\frac{\partial \ell(\theta; Y)}{\partial \theta} \right)^2 \right]}$$

- On a démontré dans **le calcul de la variance pour une densité faisant partie de la famille exponentielle que** que les deux définitions sont équivalentes et ceci est vrai pour n'importe quelle densité $f_Y(y; \theta)$.

Fonction de Score et Information de Fisher

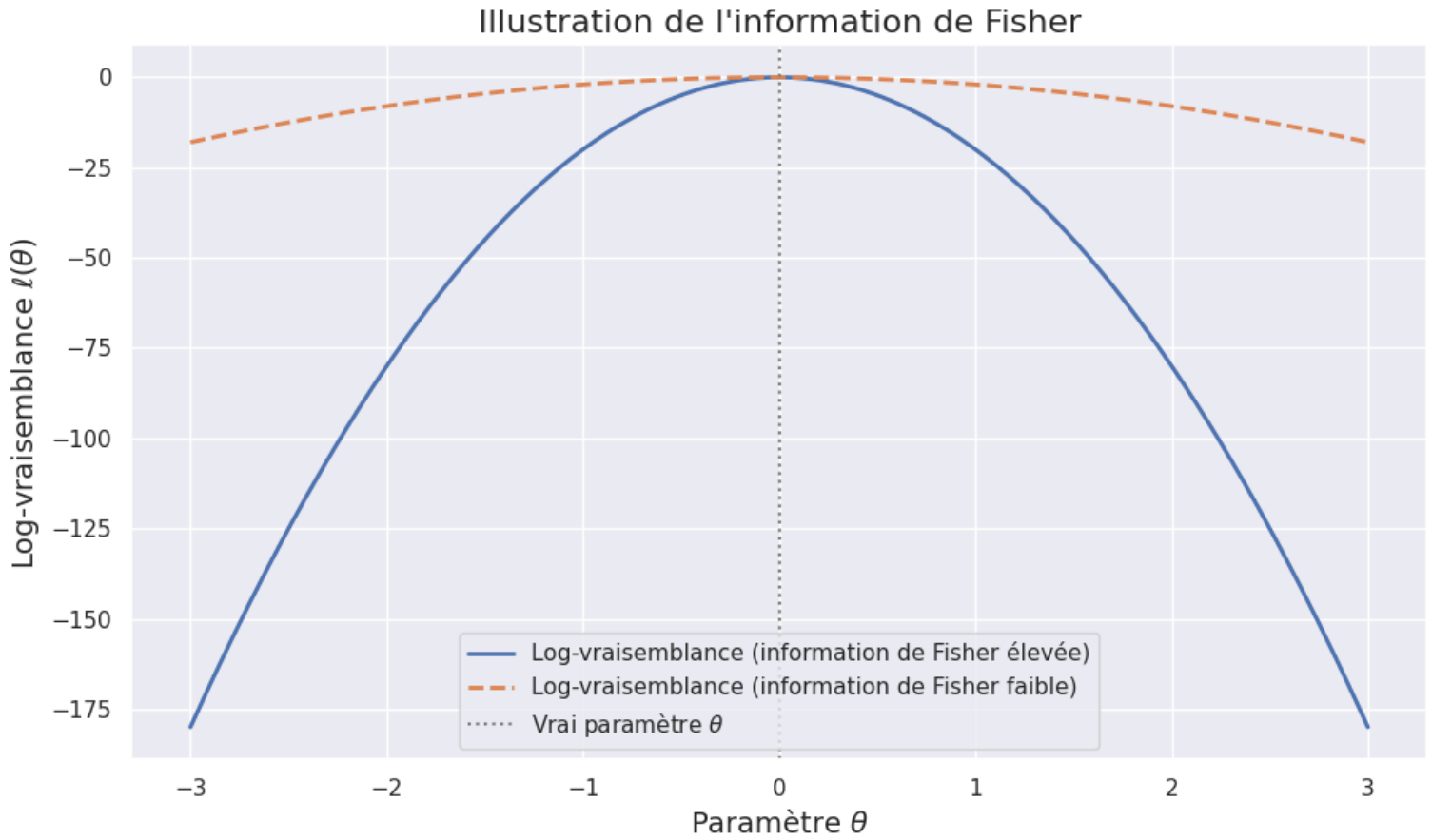
- Étant donné que $\mathbb{E}\left[\frac{\partial \ell}{\partial \theta}\right] = 0$, l'information de Fisher exprime la variance du score $\text{Var}[U(\theta)]$ puisque :

$$\begin{aligned}
\text{Var}\left(\underbrace{\frac{\partial \ell(\theta; Y)}{\partial \theta}}_{U(\theta)}\right) &= \mathbb{E}_\theta\left[\left(\frac{\partial \ell(\theta; Y)}{\partial \theta}\right)^2\right] - \left(\underbrace{\mathbb{E}_\theta\left[\frac{\partial \ell(\theta; Y)}{\partial \theta}\right]}_{0}\right)^2 \\
&= \mathbb{E}_\theta\left[\left(\frac{\partial \ell(\theta; Y)}{\partial \theta}\right)^2\right] \\
&= \mathbb{E}_\theta\left[J(\theta)\right] \\
&= I(\theta)
\end{aligned}$$

- **Interprétation intuitive :**
  - $I(\theta)$ mesure la précision moyenne de notre estimation.
  - Plus $I(\theta)$ est grand, plus la log-vraisemblance est "pointue", et plus on estime θ précisément.

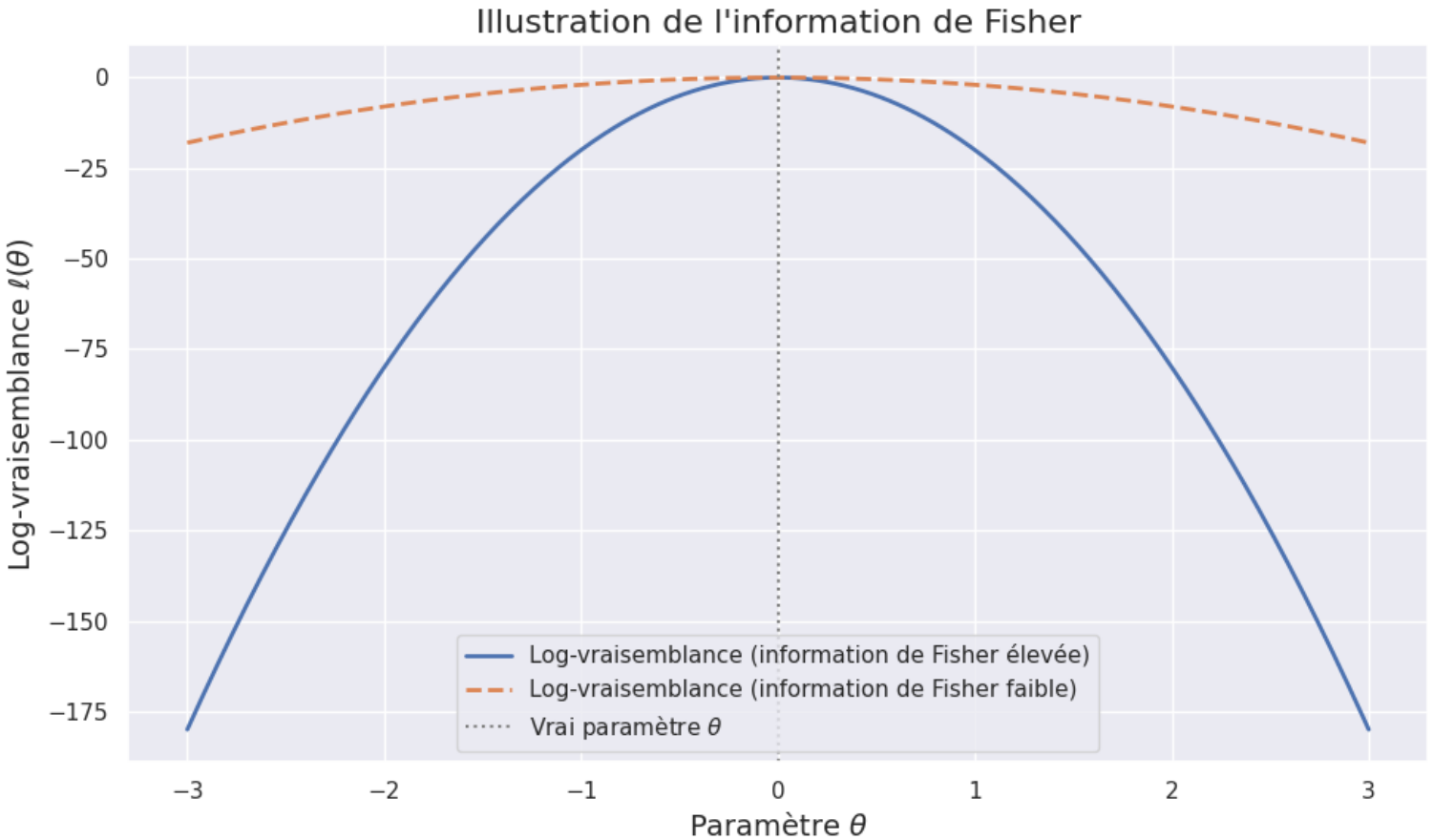
Fonction de Score et Information de Fisher

- **Interprétation intuitive :**
  - $I(\theta)$ mesure la précision moyenne de notre estimation.
  - Plus $I(\theta)$ est grand, plus la log-vraisemblance est "pointue", et plus on estime θ précisément.



Fonction de Score et Information de Fisher

- **Interprétation intuitive :**
  - Exemple : En régression linéaire où $Y \sim \mathcal{N}(\mu, \sigma^2)$, $\mathcal{I}(\theta) = -\mathbb{E}\left[\frac{\partial^2 \ell}{\partial \theta^2}\right] = \frac{b''(\theta)}{a(\phi)} = \frac{1}{\sigma^2}$. Plus le bruit est faible, plus l'estimation est précise.



Fonction de Score et Information de Fisher

Terme	**Expression**	**Interprétation**
Fonction de score $U(\theta)$	$\frac{\partial}{\partial \theta} \log f_Y(y;\theta)$	Sensibilité de la log-vraisemblance
Information observée $J(\theta)$	$-\frac{\partial^2}{\partial \theta^2} \log f_Y(y;\theta)$	Courbure locale
Information de Fisher $I(\theta)$	$\mathbb{E}[J(\theta)] = \text{Var}(U(\theta))$	Précision moyenne de l'estimation de θ

Dérivation de Solution pour le Problème de PCA

Métriques d'Évalutation et Outils de Visualisation : PCA

- On cherche une direction $\mathbf{u}_1 \in \mathbb{R}^p$ (tel que $\|\mathbf{u}_1\|^2 = 1$) dans laquelle la variance projetée des données est maximale.
- Cette variance est donnée par le produit quadratique :

$$\text{Var}(\mathbf{X}\mathbf{u}_1) = \mathbf{u}_1^\top \boldsymbol{\Sigma} \mathbf{u}_1$$

- Sous la contrainte :

$$\|\mathbf{u}_1\|^2 = \mathbf{u}_1^\top \mathbf{u}_1 = 1$$

- C'est un problème classique d'optimisation quadratique sous contrainte quadratique, que l'on résout à l'aide des multiplicateurs de Lagrange.
- **1. Formule du Lagrangien :**

$$\mathcal{L}(\mathbf{u}, \lambda) = \mathbf{u}^\top \boldsymbol{\Sigma} \mathbf{u} - \lambda(\mathbf{u}^\top \mathbf{u} - 1).$$

- **2. Condition d'optimalité :**

$$\nabla_{\mathbf{u}} \mathcal{L} = 2\boldsymbol{\Sigma}\mathbf{u} - 2\lambda\mathbf{u} = 0 \quad \Rightarrow \quad \boldsymbol{\Sigma}\mathbf{u} = \lambda\mathbf{u}.$$

Métriques d'Évalutation et Outils de Visualisation : PCA

- **1. Formule du Lagrangien :**

$$\mathcal{L}(\mathbf{u}, \lambda) = \mathbf{u}^\top \boldsymbol{\Sigma} \mathbf{u} - \lambda(\mathbf{u}^\top \mathbf{u} - 1)$$

- **2. Condition d'optimalité :**

$$\nabla_{\mathbf{u}} \mathcal{L} = 2\boldsymbol{\Sigma}\mathbf{u} - 2\lambda\mathbf{u} = 0 \quad \Rightarrow \quad \boldsymbol{\Sigma}\mathbf{u} = \lambda\mathbf{u}$$

- Cela correspond exactement à une équation aux valeurs propres.
- Donc les directions $\mathbf{u}$ qui maximisent (ou minimisent) la variance sont les vecteurs propres de $\boldsymbol{\Sigma}$.
- Il reste à vérifier la dérivée seconde (Hessienne par rapport à $\mathbf{u}$) est :

$$\nabla^2_{\mathbf{u}} \mathcal{L} = 2(\boldsymbol{\Sigma} - \lambda \mathbf{I})$$

Métriques d'Évalutation et Outils de Visualisation : PCA

- Cette Hessienne représente une forme quadratique qui décrit la courbure de la fonction dans un espace contraint : celui de la sphère unité $\|\mathbf{u}\| = 1$.
- Puisque $\boldsymbol{\Sigma}$ est symétrique, elle est diagonalisable, et ses vecteurs propres forment une base orthonormée.
- On peut donc analyser la nature du point stationnaire à partir des valeurs propres de la matrice $\boldsymbol{\Sigma} - \lambda \mathbf{I}$ projetée sur l'espace tangent à la sphère.

Métriques d'Évalutation et Outils de Visualisation : PCA

- Soit $\mathbf{u}_1$ un vecteur propre de $\boldsymbol{\Sigma}$ associé à la plus grande valeur propre λ_1.
- Toute direction $\mathbf{v}$ dans l'espace tangent à la sphère en $\mathbf{u}_1$ vérifie $\mathbf{v}^\top \mathbf{u}_1 = 0$, donc peut s'écrire :

$$\mathbf{v} = \sum_{i=2}^{p} \alpha_i \mathbf{u}_i$$

- On évalue la courbure dans cette direction :

$$\mathbf{v}^\top \nabla^2_{\mathbf{u}} \mathcal{L}\, \mathbf{v} = 2 \sum_{i=2}^{p} \alpha_i^2 (\lambda_i - \lambda_1) < 0$$

- Donc la Hessienne est négative définie sur l'espace tangent : $\mathbf{u}_1$ est bien un maximum local.

Métriques d'Évalutation et Outils de Visualisation : PCA

- La Hessienne du Lagrangien est :

$$\nabla^2_{\mathbf{u}}\mathcal{L} = 2(\boldsymbol{\Sigma} - \lambda\mathbf{I})$$

- Elle est négative semi-définie dans l'espace contraint si $\lambda = \lambda_1$, la plus grande valeur propre.
- Cela confirme que $\mathbf{u}_1$ est bien un maximum.
- Pour toute direction orthogonale à $\mathbf{u}_1$, la variance projetée est strictement inférieure à la valeur propre associée à $\mathbf{u}_1$:

$$\forall \mathbf{v} \perp \mathbf{u}_1, \quad \mathbf{v}^\top \boldsymbol{\Sigma} \mathbf{v} \leq \lambda_2 < \lambda_1$$

- La fonction est donc localement concave autour de $\mathbf{u}_1$ sur la sphère unité.

Métriques d'Évalutation et Outils de Visualisation : PCA

- Si $\mathbf{u}_1$ est un vecteur propre associé à la plus grande valeur propre λ_1, alors :
$$\nabla^2_{\mathbf{u}}\mathcal{L} = 2(\boldsymbol{\Sigma} - \lambda_1 \mathbf{I})$$
est négative semi-définie sur l'espace tangent à la contrainte (les directions orthogonales à $\mathbf{u}_1$).
- Ainsi, $\mathbf{u}_1$ est bien un maximum local (et comme la valeur atteinte est la plus grande possible, il s'agit d'un maximum global) de la variance projetée sous la contrainte $\|\mathbf{u}\| = 1$.
- Les valeurs prises par $\mathbf{u}^\top \boldsymbol{\Sigma} \mathbf{u}$ sont exactement les valeurs propres λ_i lorsque $\mathbf{u}$ est un vecteur propre.
- Le maximum est donc atteint lorsque $\mathbf{u}_1$ est le vecteur propre associé à la plus grande valeur propre λ_1.

Résumé

Chercher la direction qui maximise la variance projetée revient à résoudre un problème aux valeurs propres sur la matrice de covariance $\boldsymbol{\Sigma}$. La direction optimale est le vecteur propre principal, car c'est celle où les données sont le plus étalées.