

Hicham Janati

[hjanati@insea.ac.ma](mailto:hjanati@insea.ac.ma)



# Chapitre 3. Applications et thématiques avancées

1. Modèles Bayésiens hiérarchiques
2. Classical Machine learning: zero to hero
3. Bayesian Machine learning

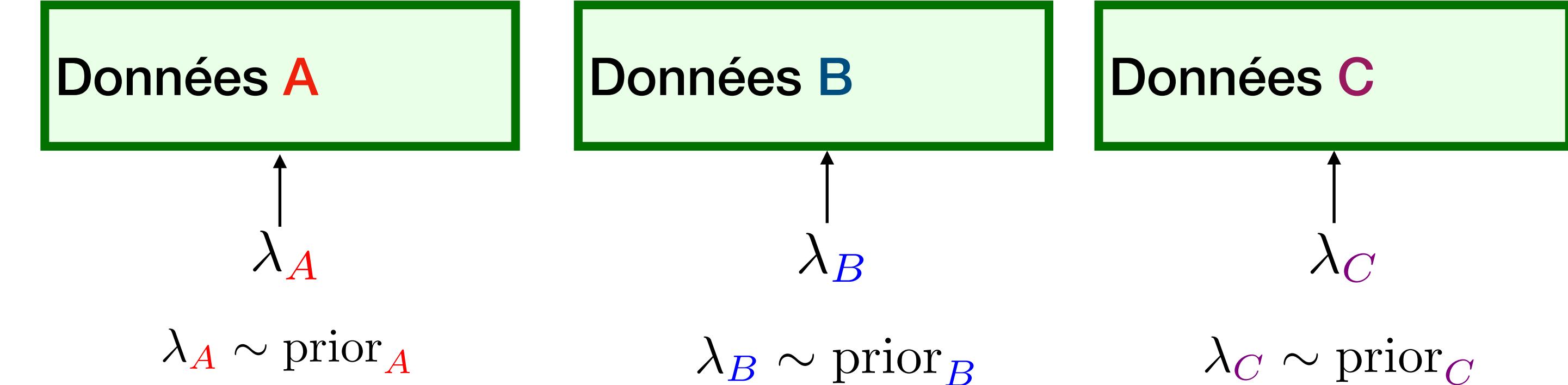
# Modèles Bayésiens hiérarchiques

On souhaite modéliser la fréquence des sinistres d'un ensemble de conducteurs dans trois villes différentes A, B, C

Trois variables à estimer:  $\lambda_A, \lambda_B, \lambda_C$

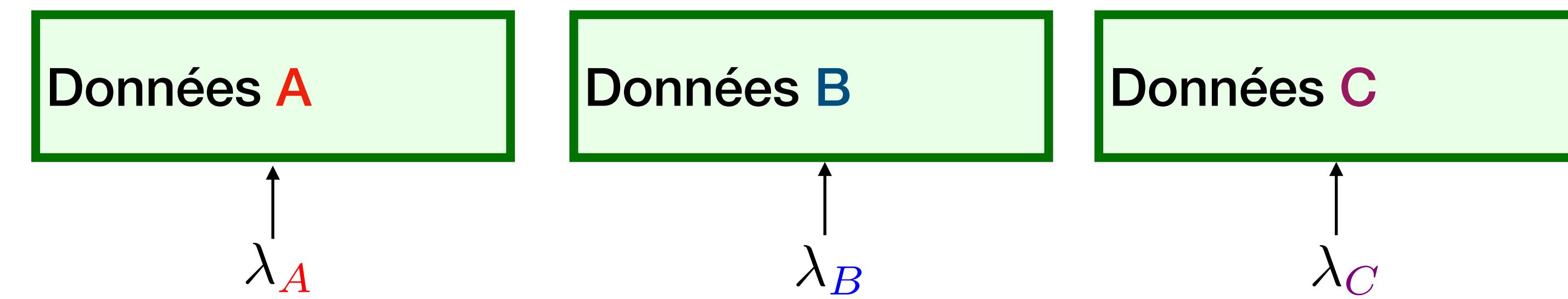
On peut considérer une approche indépendante:

Quels sont les **inconvénients** de ce modèle ?



Aucun lien entre les régions: on n'exploite pas les similarités entre les régions

Et si on utilise la même prior ?



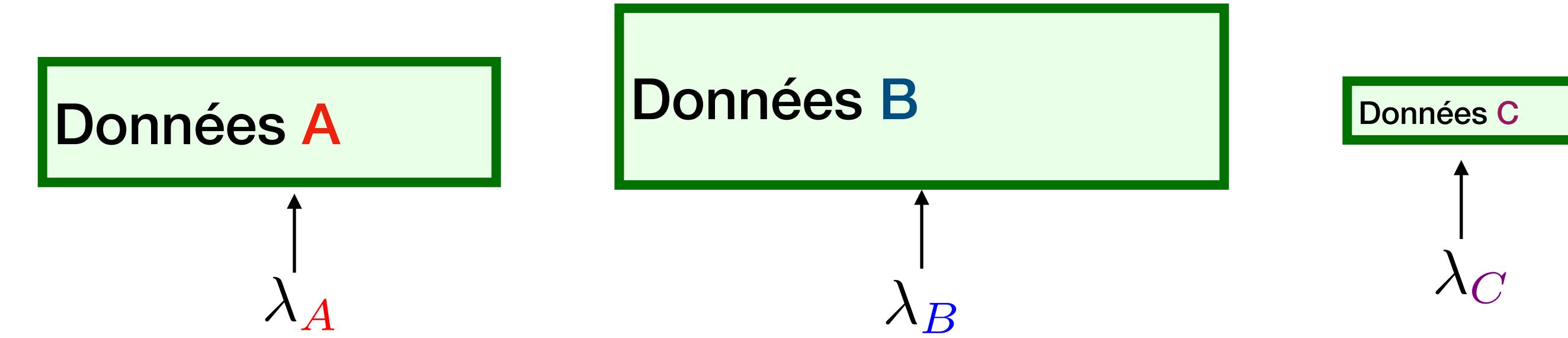
Implicitement à quoi correspondent les quantités:

$$\frac{\alpha}{\beta} \text{ et } \frac{\alpha}{\beta^2}$$

$$\lambda_A, \lambda_B, \lambda_C \sim \text{Gamma}(\alpha, \beta)$$

$\alpha, \beta$  fixés (vaguement, ou données historiques)

# Modèles Bayésiens hiérarchiques



Ne pas forcer les paramètres a priori, les considérer comme des variables aléatoires à estimer:

prior       $\lambda_A, \lambda_B, \lambda_C \sim \text{Gamma}(\alpha, \beta)$

hyperprior       $\alpha \sim \text{prior}(a)$        $\beta \sim \text{prior}(b)$

$a, b$  fixés (vaguement, données historiques)

Un modèle bayésien hiérarchique modélise les similarités et les différences entre les groupes à partir des données

Données de mortalité dans des hôpitaux américains.

| YEAR | HOSPITAL          | Procedure/Condition       | # of Deaths | # of Cases |
|------|-------------------|---------------------------|-------------|------------|
| 2016 | Highland Hospital | Acute Stroke              | 17          | 147        |
| 2016 | Highland Hospital | Acute Stroke Hemorrhagic  | 10          | 36         |
| 2016 | Highland Hospital | Acute Stroke Ischemic     | 6           | 106        |
| 2016 | Highland Hospital | Acute Stroke Subarachnoid | 1           | 5          |
| 2016 | Highland Hospital | Carotid Endarterectomy    | 0           | 5          |
| 2016 | Highland Hospital | Espophageal Resection     | 0           | 3          |
| 2016 | Highland Hospital | GI Hemorrhage             | 4           | 147        |
| 2016 | Highland Hospital | Heart Failure             | 1           | 317        |
| 2016 | Highland Hospital | Hip Fracture              | 1           | 38         |
| 2016 | Highland Hospital | PCI                       | 10          | 132        |

1. Vous êtes data scientist.
2. Votre tâche est vague: “on veut un rapport sur les hôpitaux dans le pays”
3. Que faites-vous ?

# Modèles Bayésiens hiérarchiques

## Application

| YEAR | HOSPITAL          | Procedure/Condition       | # of Deaths | # of Cases |
|------|-------------------|---------------------------|-------------|------------|
| 2016 | Highland Hospital | Acute Stroke              | 17          | 147        |
| 2016 | Highland Hospital | Acute Stroke Hemorrhagic  | 10          | 36         |
| 2016 | Highland Hospital | Acute Stroke Ischemic     | 6           | 106        |
| 2016 | Highland Hospital | Acute Stroke Subarachnoid | 1           | 5          |
| 2016 | Highland Hospital | Carotid Endarterectomy    | 0           | 5          |
| 2016 | Highland Hospital | Espophageal Resection     | 0           | 3          |

### 1. Problématiques simples :

1. Classement des hôpitaux par taux de mortalité
2. Classement des procédures par taux de mortalité
3. Classement des hôpitaux + procédures par taux de mortalité
4. Étudier l'évolution des taux de mortalité dans le temps

### 2. Statistiques descriptives:

1. Combien y a-t-il d'hôpitaux ? de procédures ? d'années ?
2. Données manquantes / dupliquées ?
3. Calculer un taux de mortalité fréquentiste.
4. Visualiser les hôpitaux / procédures avec une ACP.
5. Clusters évidents ? Outliers ?

### 3. Modélisation bayésienne

1. Pourquoi ne pas se contenter des taux fréquentistes ?
2. Définir les groupes et les lois a priori
3. Interpréter les taux de mortalité avec leur HDI

### 4. Expliquer ces données avec des données externes

1. Données géographiques (ville / quartier de l'hôpital)
2. Données par hôpital (effectif, technologies utilisées, reviews)
3. Données temporelles (événements rares: accidents, pandémies..)

# Chapitre 3. Applications et thématiques avancées

1. Modèles Bayésiens hiérarchiques (Assurance / Biostats)
2. Classical Machine learning: zero to hero
3. Bayesian Machine learning

Un opérateur téléphonique a les données historiques sur ses clients.

| Dependents | TechSupport | Contract | InternetService | Months | MonthlyCharges | Churn |
|------------|-------------|----------|-----------------|--------|----------------|-------|
| 0          | 1           | 0        | 1               | 12     | 75.65          | 0     |
| 1          | 0           | 0        | 0               | 24     | 89.50          | 0     |
| 0          | 0           | 0        | 1               | 6      | 65.25          | 1     |
| 0          | 1           | 1        | 0               | 48     | 35.30          | ?     |
| 1          | 0           | 0        | 1               | 48     | 85.81          | ?     |

Churn = 1: client a annulé son abonnement

L'entreprise souhaite anticiper le “churn” avec un algorithme de prédiction pour cibler les clients concernés

$$\mathbf{X} = (\mathbf{X}^1, \dots, \mathbf{X}^6) \rightarrow \mathbf{y} \in \{0, 1\}$$

On cherche une fonction  $f$  telle que:  $f(\mathbf{X}) \approx \mathbf{y}$

$$\min_f \sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2 \quad \text{Erreur de prédiction}$$

$f$  doit donner 1 ou 0, on considère alors des fonctions de type:  $f(\mathbf{x}) = \mathbb{1}_{g(\mathbf{x}) \geq 0}$

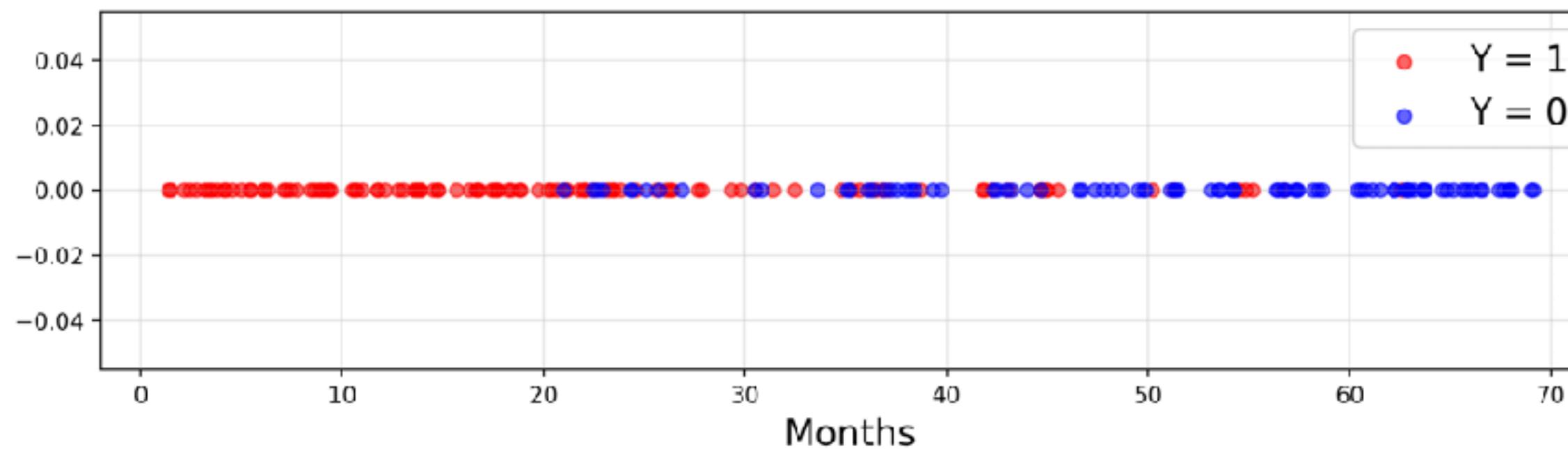
On ne peut pas chercher  $g$  dans la totalité de l'espace des fonctions (dimension infinie), il faut paramétriser  $g$

$f$  doit donner 1 ou 0, on considère alors des fonctions de type:  $f(\mathbf{x}) = \mathbb{1}_{g(\mathbf{x}) \geq 0}$

On ne peut pas chercher  $g$  dans la totalité de l'espace des fonctions (dimension infinie), il faut paramétriser  $g$

On considère une seule variable “Months” qui donne la durée du contrat:

$$\mathbf{x} = \text{Months} \in \mathbb{R}$$



Quelle serait la fonction paramétrée  $g$  la plus simple ici ?

$$g(\mathbf{x}) = \beta_1 \mathbf{x} + \beta_0, \quad \beta_0, \beta_1 \in \mathbb{R}$$

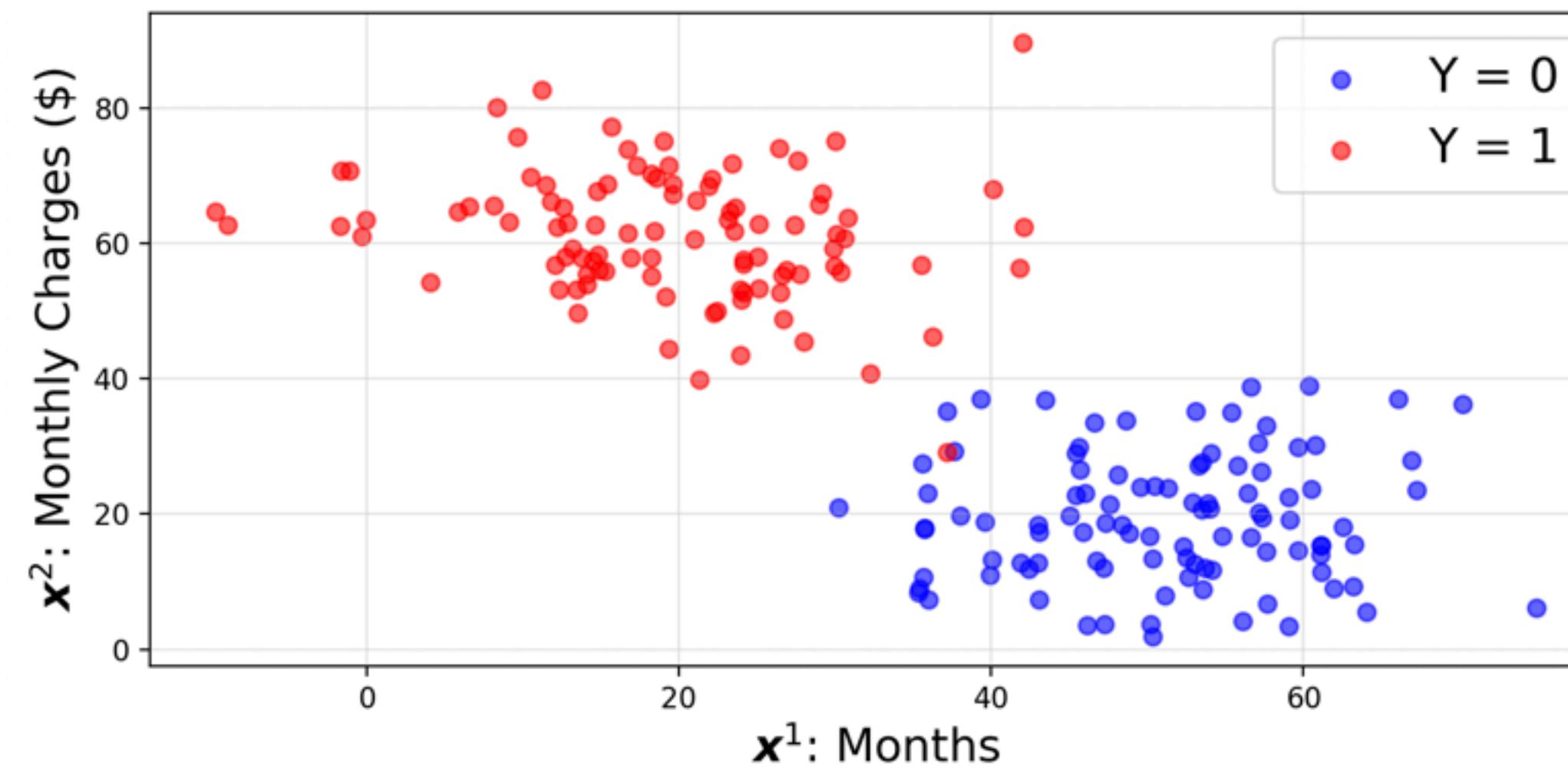
Chercher la meilleure  $f$  = chercher le meilleur  $\beta$ :

$$\min_{\beta \in \mathbb{R}^2} \sum_{i=1}^n (\mathbb{1}_{\{\beta_1 \mathbf{x}_i + \beta_0 \geq 0\}} - y_i)^2$$

Pouvez-vous donner des estimations vagues de ces paramètres ?

# Machine learning classique: zero-to-hero

On considère une deux variables: “Months” et “MonthlyCharges”:



## séparateur linéaire en dimension 2

$$\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2) \quad f(\mathbf{x}) = \mathbb{1}_{g(\mathbf{x}) \geq 0}$$

Quelle serait la fonction paramétrée  $g$  la plus simple ici ?

$$g(\mathbf{x}) = \alpha + \beta_1 \mathbf{x}^1 + \beta_2 \mathbf{x}^2, \quad \alpha, \beta_1, \beta_2 \in \mathbb{R}$$

$$g(\mathbf{x}) = \alpha + \langle \beta, \mathbf{x} \rangle, \quad \alpha \in \mathbb{R}, \beta \in \mathbb{R}^2$$

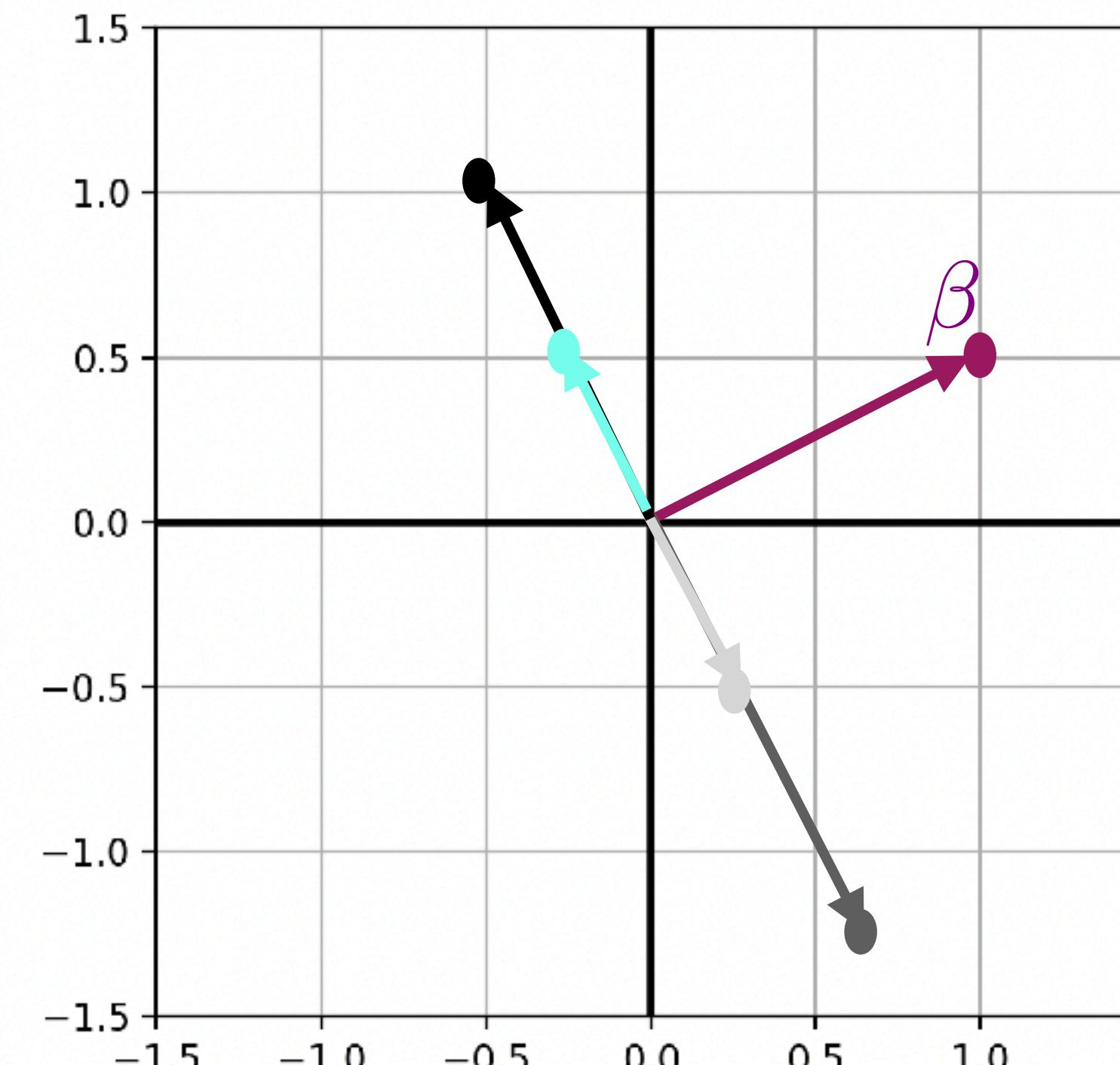
$$g(\mathbf{x}) = \alpha + \beta^\top \mathbf{x}, \quad \alpha \in \mathbb{R}, \beta \in \mathbb{R}^2$$

$$\min_{\alpha \in \mathbb{R}, \beta \in \mathbb{R}^2} \sum_{i=1}^n (\mathbb{1}_{\{\alpha + \beta^\top \mathbf{x}_i \geq 0\}} - y_i)^2$$

À quoi ressemble l’ensemble des fonctions  $g$  ?

On considère  $g : \mathbf{x} \mapsto \beta^\top \mathbf{x}$ . Étudions ses courbes de niveaux, c-à-d pour  $c \in \mathbb{R}$  les ensembles:  $\{\mathbf{x} | g(\mathbf{x}) = c\}$ .

On considère  $\mathbf{g} : \mathbf{x} \mapsto \boldsymbol{\beta}^\top \mathbf{x}$ . Étudions ses courbes de niveaux, c-à-d pour  $c \in \mathbb{R}$  les ensembles:  $\{\mathbf{x} | \mathbf{g}(\mathbf{x}) = c\}$ .



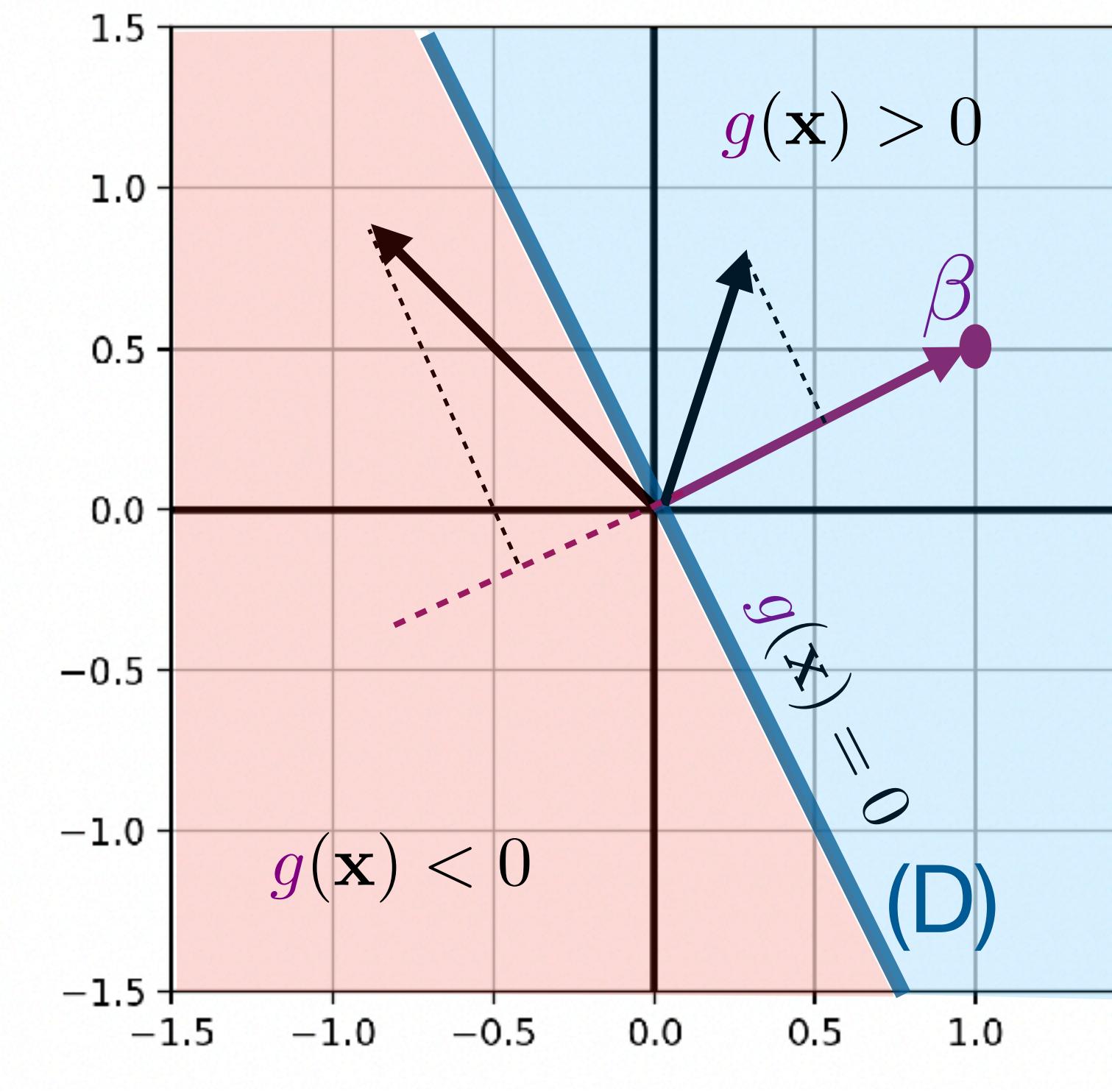
Exemple avec  $\boldsymbol{\beta} = (1, 0.5)^\top$  et  $c = 0$ .

Quels sont les  $\mathbf{x}$  tels que  $\boldsymbol{\beta}^\top \mathbf{x} = 0$  ?

Tous les vecteurs orthogonaux à  $\boldsymbol{\beta}$ .

$\{\mathbf{x} \in \mathbb{R}^2 | \boldsymbol{\beta}^\top \mathbf{x} = 0\}$  est la droite perpendiculaire à  $\boldsymbol{\beta}$ .

On considère  $g : \mathbf{x} \mapsto \beta^\top \mathbf{x}$ . Étudions ses courbes de niveaux, c-à-d pour  $c \in \mathbb{R}$  les ensembles:  $\{\mathbf{x} | g(\mathbf{x}) = c\}$ .



et si  $c = 1$  ? ou  $c = -1$  ?

Exemple avec  $\beta = (1, 0.5)^\top$  et  $c = 0$ .

Quels sont les  $\mathbf{x}$  tels que  $\beta^\top \mathbf{x} = 0$  ?

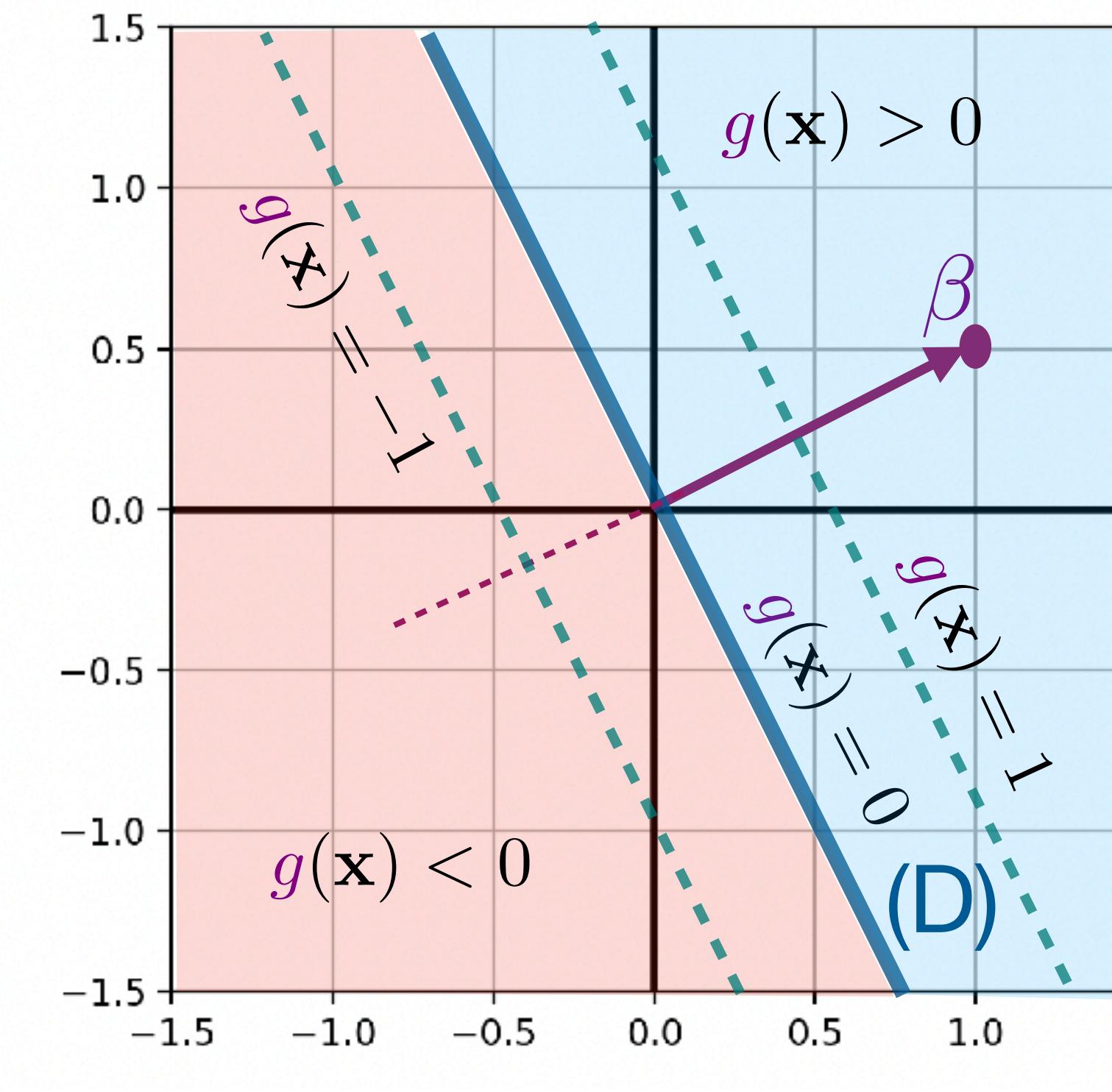
Tous les vecteurs orthogonaux à  $\beta$ .

$\{\mathbf{x} \in \mathbb{R}^2 | \beta^\top \mathbf{x} = 0\}$  est la droite perpendiculaire à  $\beta$ .

à droite de (D),  $\beta^\top \mathbf{x} > 0$

à gauche de (D),  $\beta^\top \mathbf{x} < 0$

On considère  $g : \mathbf{x} \mapsto \beta^\top \mathbf{x}$ . Étudions ses courbes de niveaux, c-à-d pour  $c \in \mathbb{R}$  les ensembles:  $\{\mathbf{x} | g(\mathbf{x}) = c\}$ .



et si  $c = 1$  ? ou  $c = -1$  ?

Exemple avec  $\beta = (1, 0.5)^\top$  et  $c = 0$ .

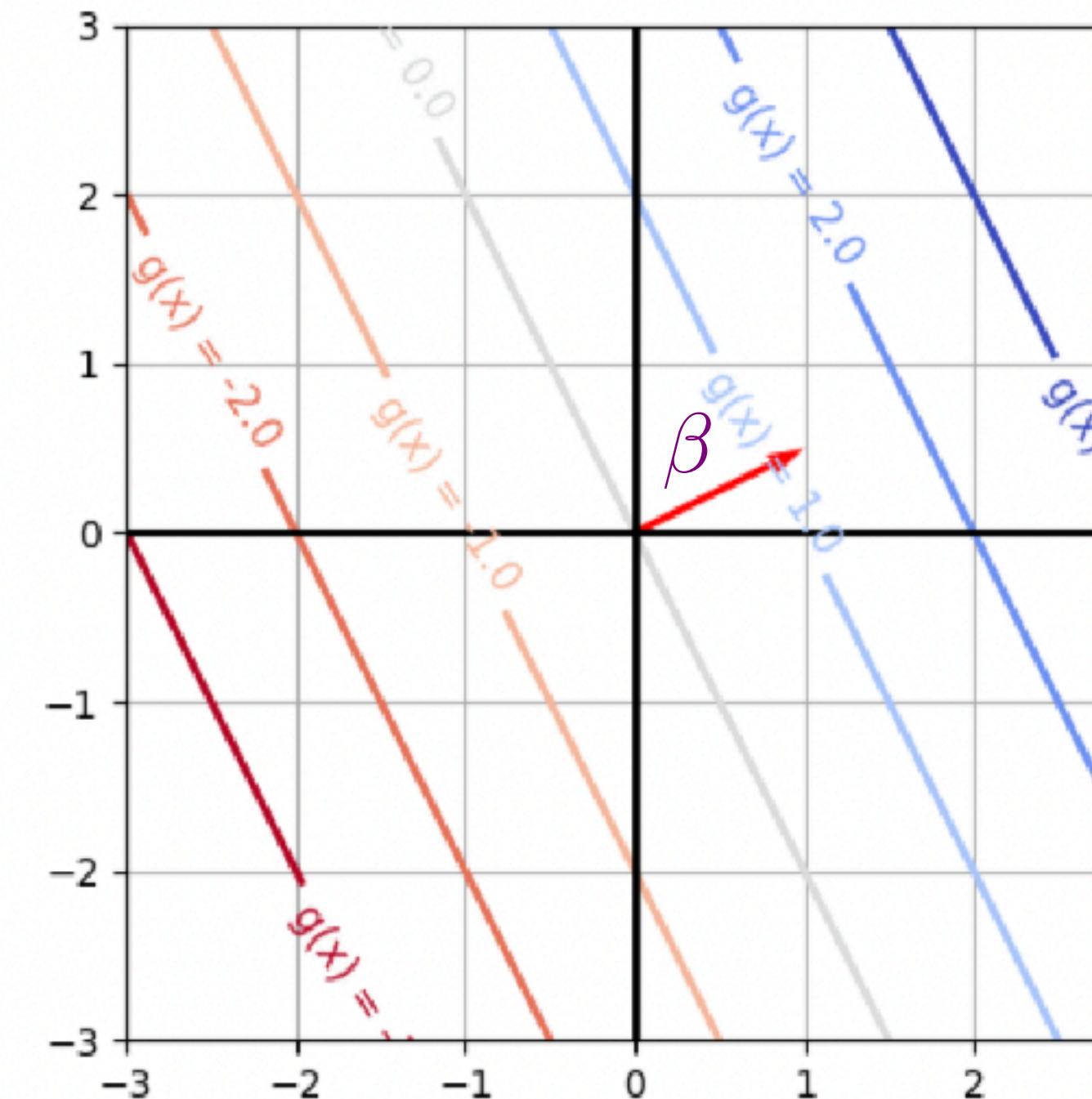
Quels sont les  $\mathbf{x}$  tels que  $\beta^\top \mathbf{x} = 0$  ?

Tous les vecteurs orthogonaux à  $\beta$ .

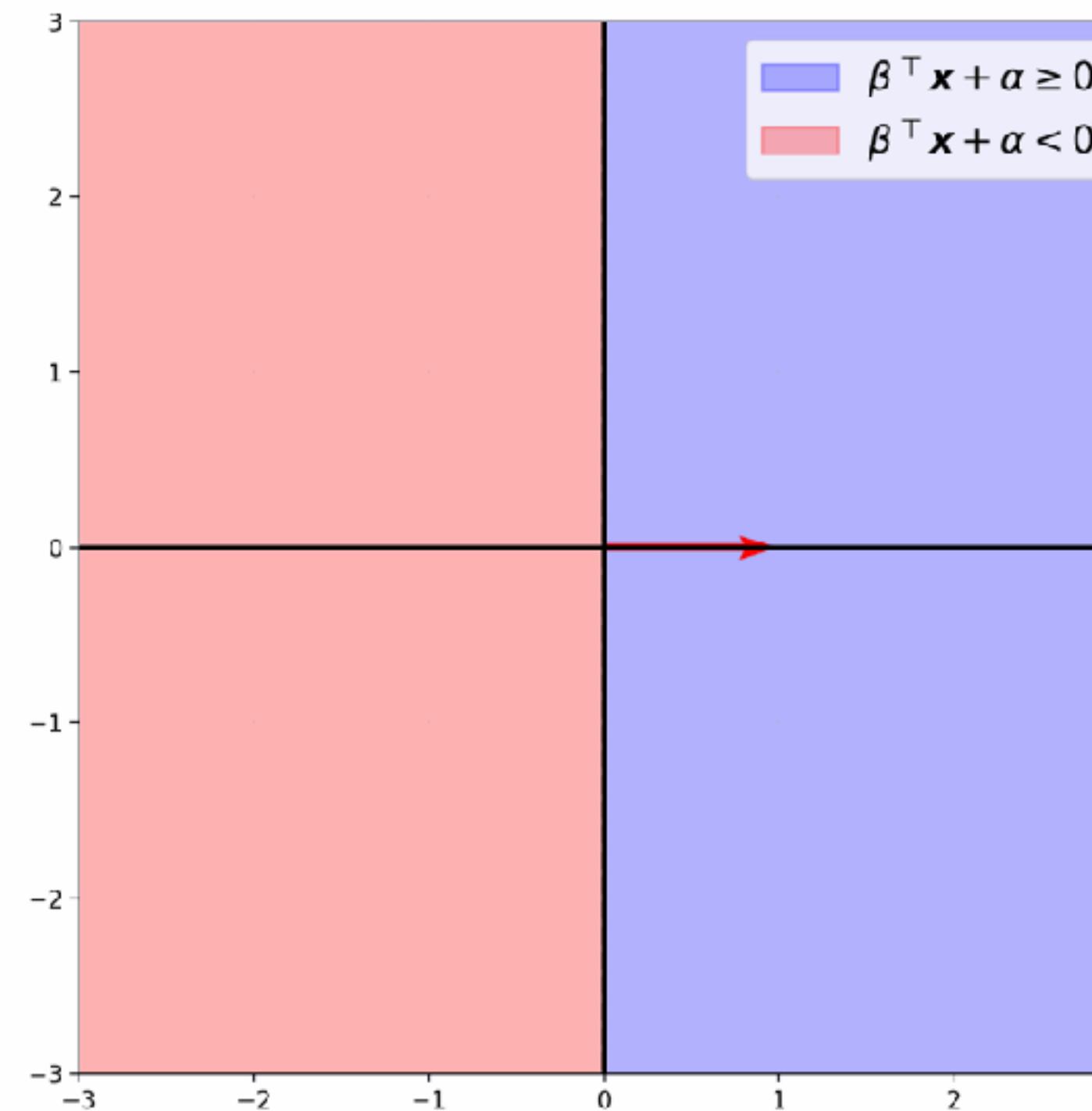
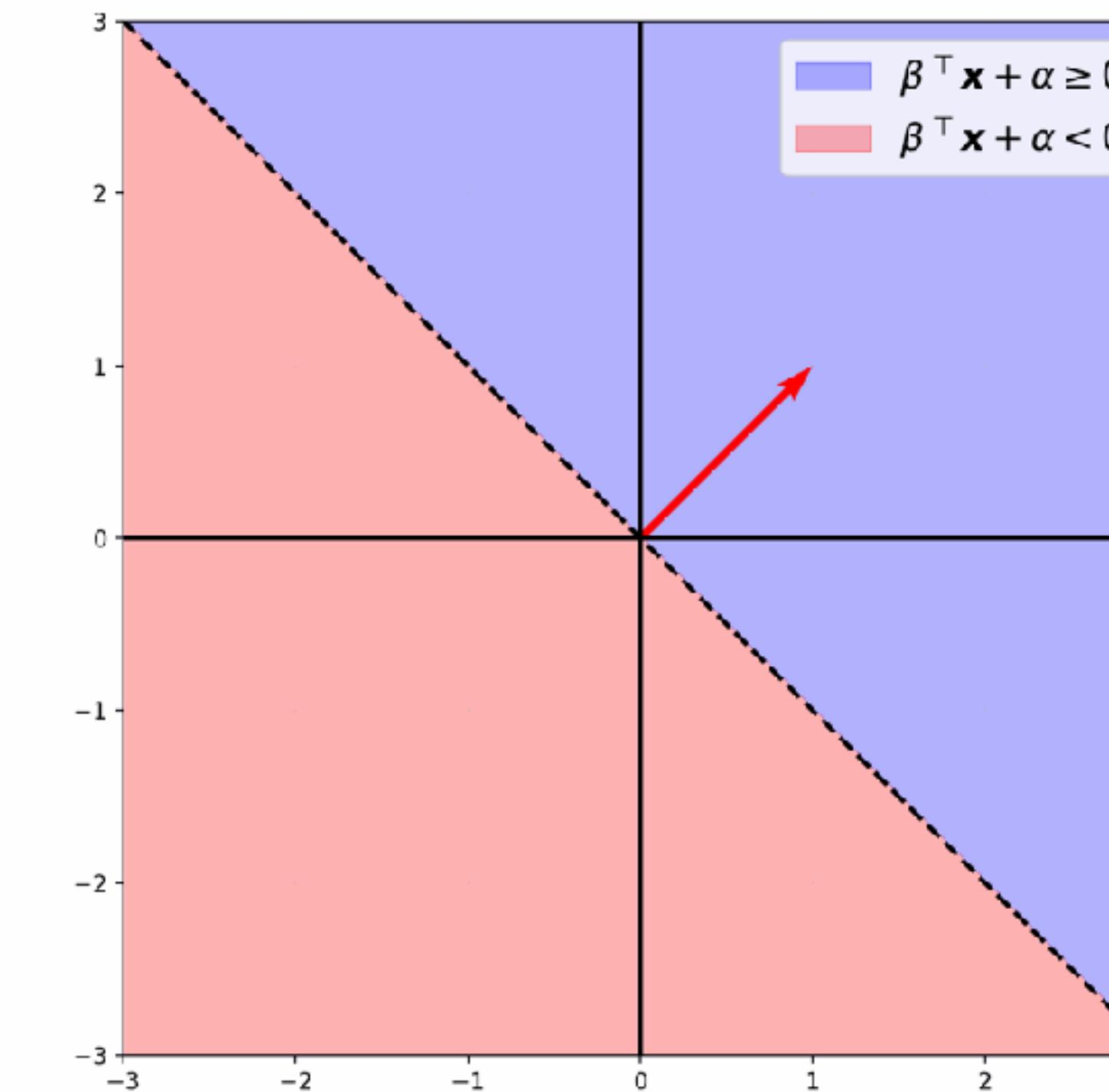
$\{\mathbf{x} \in \mathbb{R}^2 | \beta^\top \mathbf{x} = 0\}$  est la droite perpendiculaire à  $\beta$ .

à droite de (D),  $\beta^\top \mathbf{x} > 0$

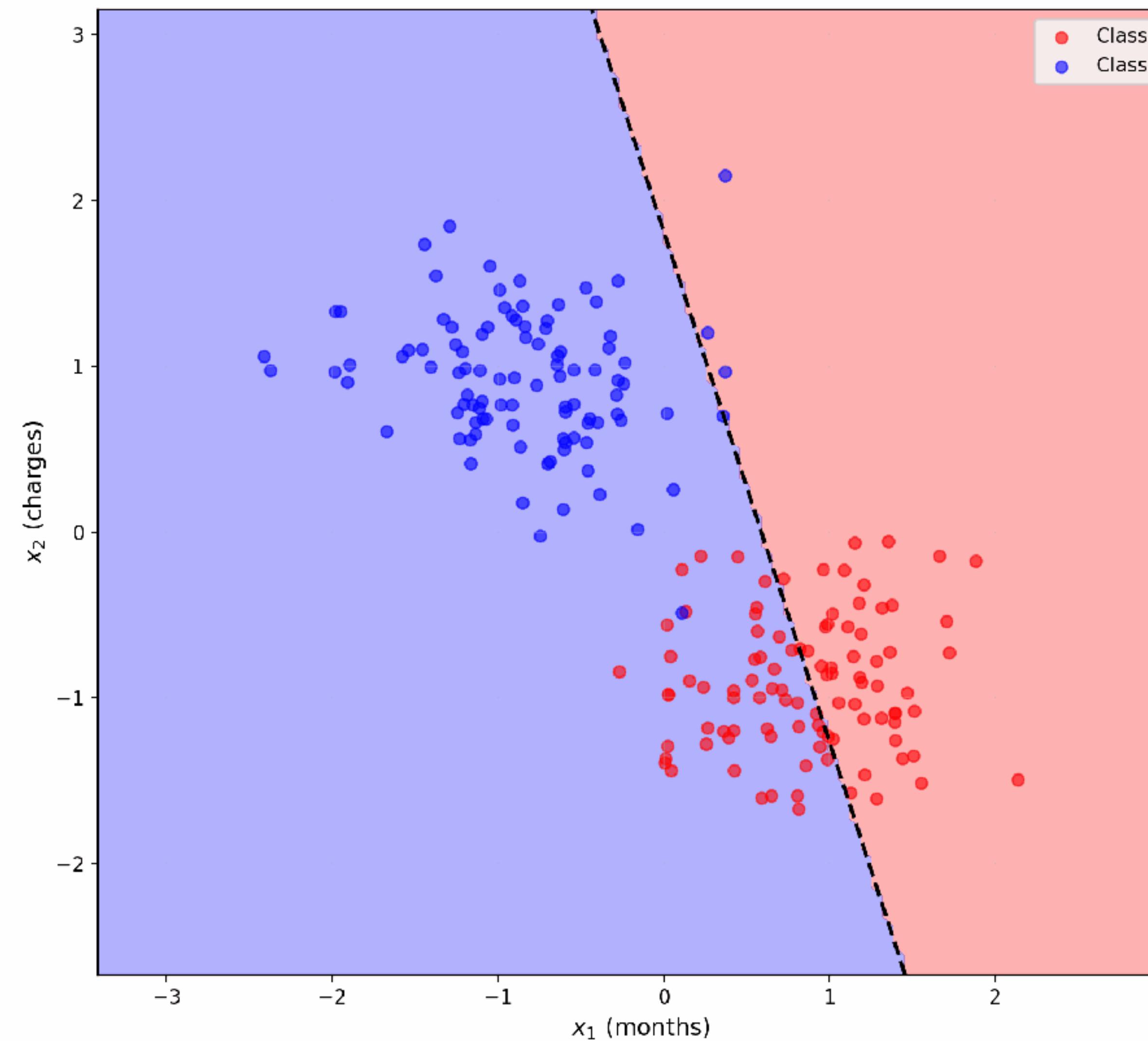
à gauche de (D),  $\beta^\top \mathbf{x} < 0$



Comment change la fonction de prédiction  $f : \mathbb{1}_{\{\alpha + \beta^\top \mathbf{x} \geq 0\}}$  en fonction de  $\alpha$  et  $\beta$  ?

$\alpha = 0, \beta$  varie: $\alpha$  varie,  $\beta = [1, 1]$ :Comment change la fonction de prédiction  $f : \mathbb{1}_{\{\alpha + \beta^\top x \geq 0\}}$  en fonction de  $\alpha$  et  $\beta$  ?

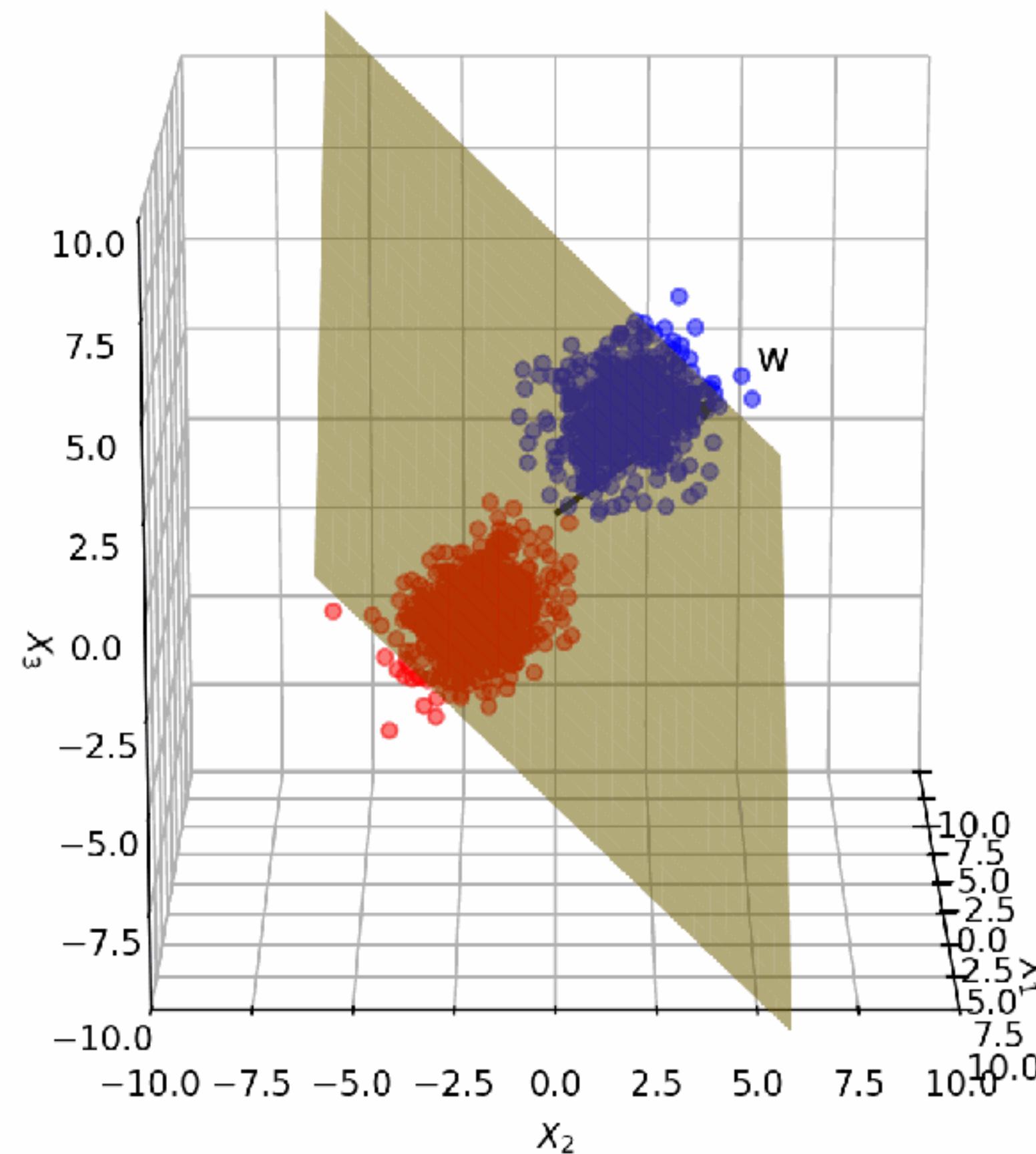
$$\min_{\alpha \in \mathbb{R}, \beta \in \mathbb{R}^2} \sum_{i=1}^n (\mathbb{1}_{\{\alpha + \beta^\top \mathbf{x}_i \geq 0\}} - y_i)^2$$



Et si on utilise trois variables:

$$\textcolor{violet}{g}(\mathbf{x}) = \alpha + \beta_1 x^1 + \beta_2 x^2 + \beta_3 x^3$$

$$\textcolor{violet}{g}(\mathbf{x}) = \alpha + \boldsymbol{\beta}^\top \mathbf{x}$$



Que forment les  $\mathbf{x}$  tels que  $\{\textcolor{violet}{g}(\mathbf{x}) = 0\}$ ?

En dimension d:  $\textcolor{violet}{g}(\mathbf{x}) = \alpha + \boldsymbol{\beta}^\top \mathbf{x}, \quad \boldsymbol{\beta} \in \mathbb{R}^d$

Que forment les  $\mathbf{x}$  tels que  $\{\textcolor{violet}{g}(\mathbf{x}) = 0\}$ ?

Un espace de dimension d-1: un hyperplan

$$\min_{\alpha \in \mathbb{R}, \beta \in \mathbb{R}^d} \sum_{i=1}^n (\mathbb{1}_{\{\alpha + \beta^\top \mathbf{x}_i \geq 0\}} - y_i)^2$$

Fonction non différentiable (discontinue même) difficile à optimiser

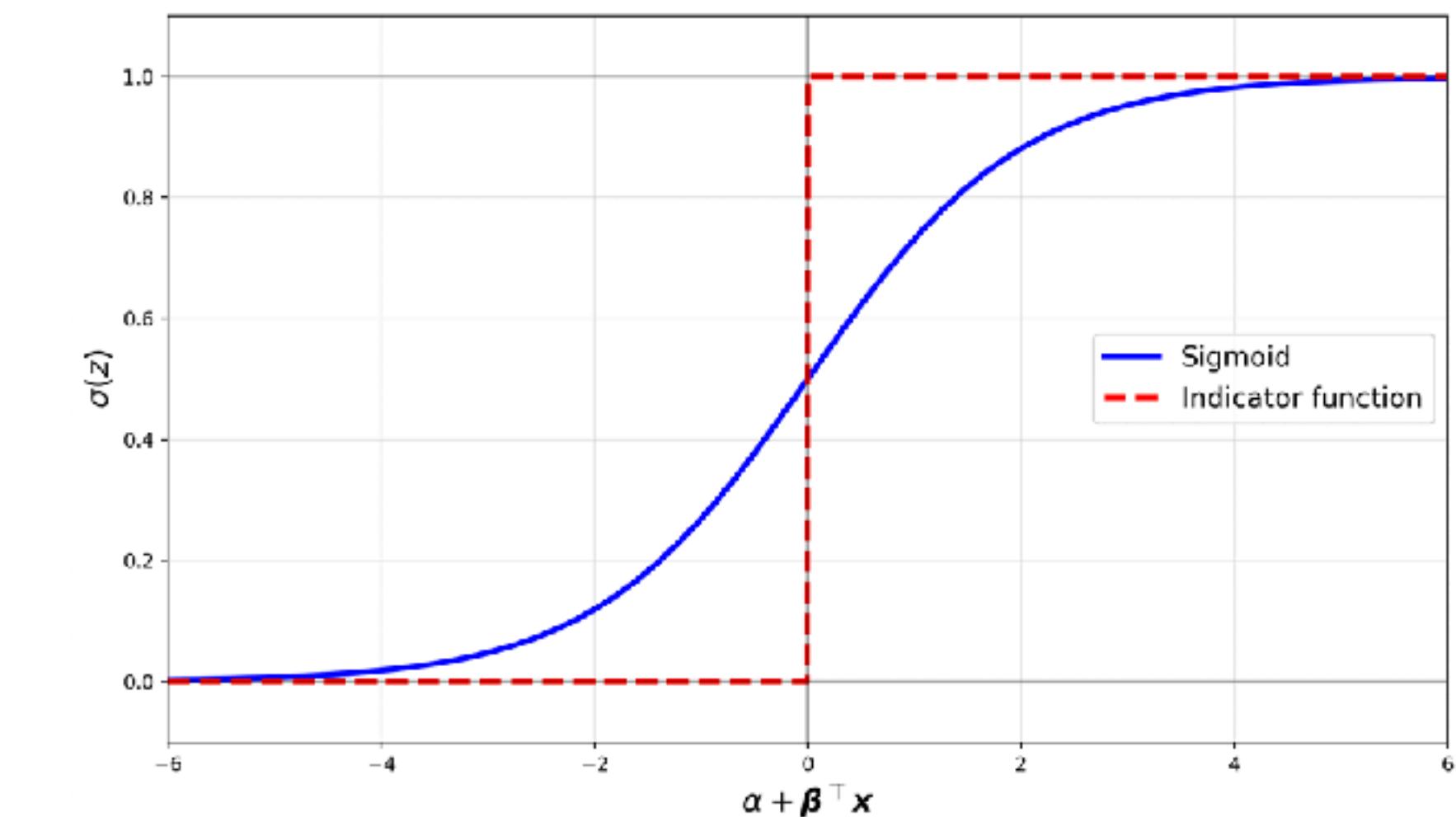
Au lieu de prendre le signe, transformer les scores  $\alpha + \beta^\top \mathbf{x}_i$  vers  $[0, 1]$  et modéliser des probabilités

sigmoid:  $t \mapsto \frac{1}{1+e^{-t}}$  (logistique)

$$p_i \stackrel{\text{def}}{=} \mathbb{P}(y_i = 1 | \mathbf{x}_i) = \text{sigmoid}(\alpha + \beta^\top \mathbf{x}_i)$$

On peut comparer les  $p_i$  avec les  $y_i$  avec la *cross-entropy*:

$$\min_{\alpha \in \mathbb{R}, \beta \in \mathbb{R}^d} - \sum_{i=1}^n y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$



On a donc une fonction de prédiction:  $f^*(\mathbf{x}_i) = 1 \Leftrightarrow \text{sigmoid}(\alpha^* + \beta^{*\top} \mathbf{x}_i) \geq \frac{1}{2}$

## Modèle de régression logistique

$$\textcolor{violet}{p}_i \stackrel{\text{def}}{=} \mathbb{P}(y_i = 1 | \mathbf{x}_i) = \text{sigmoid}(\alpha + \beta^\top \mathbf{x}_i)$$

Optimisation faite sur  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$

$$\min_{\alpha \in \mathbb{R}, \beta \in \mathbb{R}^d} - \sum_{i=1}^n y_i \log(\textcolor{violet}{p}_i) + (1 - y_i) \log(1 - \textcolor{violet}{p}_i)$$

“Training” data

| $\mathbf{x}_1$ | $y_1$    |
|----------------|----------|
| $\vdots$       | $\vdots$ |
| $\mathbf{x}_n$ | $y_n$    |

→ “Training” → “Learned”  $f^*$  →

| predictions         | true labels |
|---------------------|-------------|
| $f^*(\mathbf{x}_1)$ | $y_1$       |
| $\vdots$            | $\vdots$    |
| $f^*(\mathbf{x}_n)$ | $y_n$       |

→ “Train” error

Est-ce une bonne manière d'évaluation la performance du modèle ?

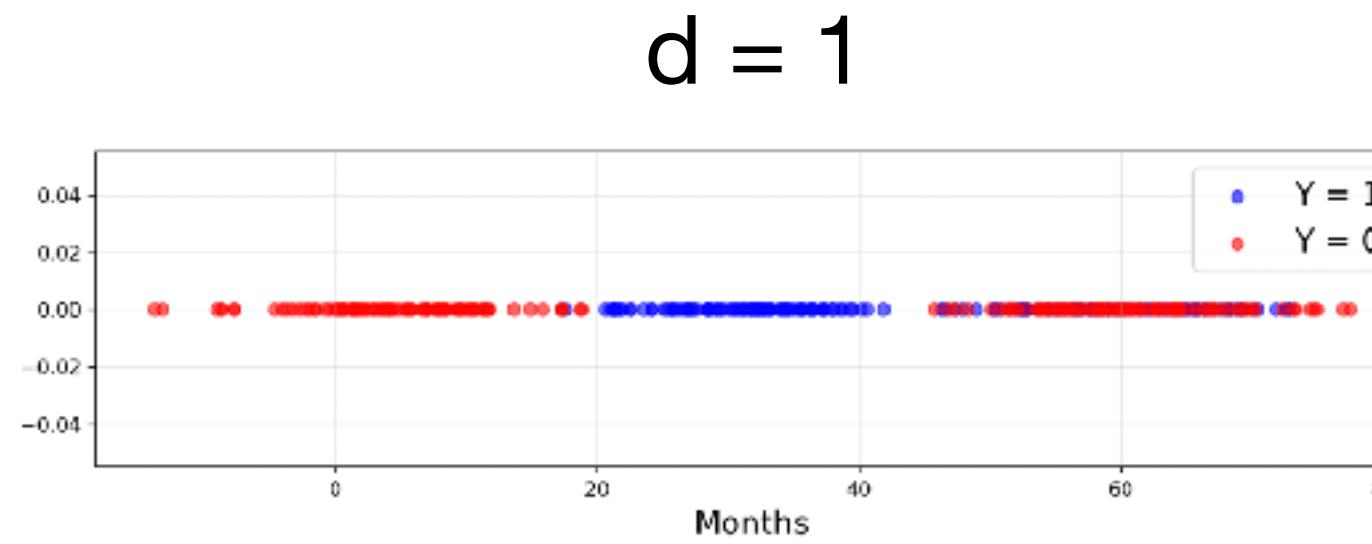
L'erreur de prédiction sur ces données est **optimisée**: elle est forcément **petite**.

Il faut évaluer la performance du modèle sur des données nouvelles non vues à l'entraînement: “Test data”

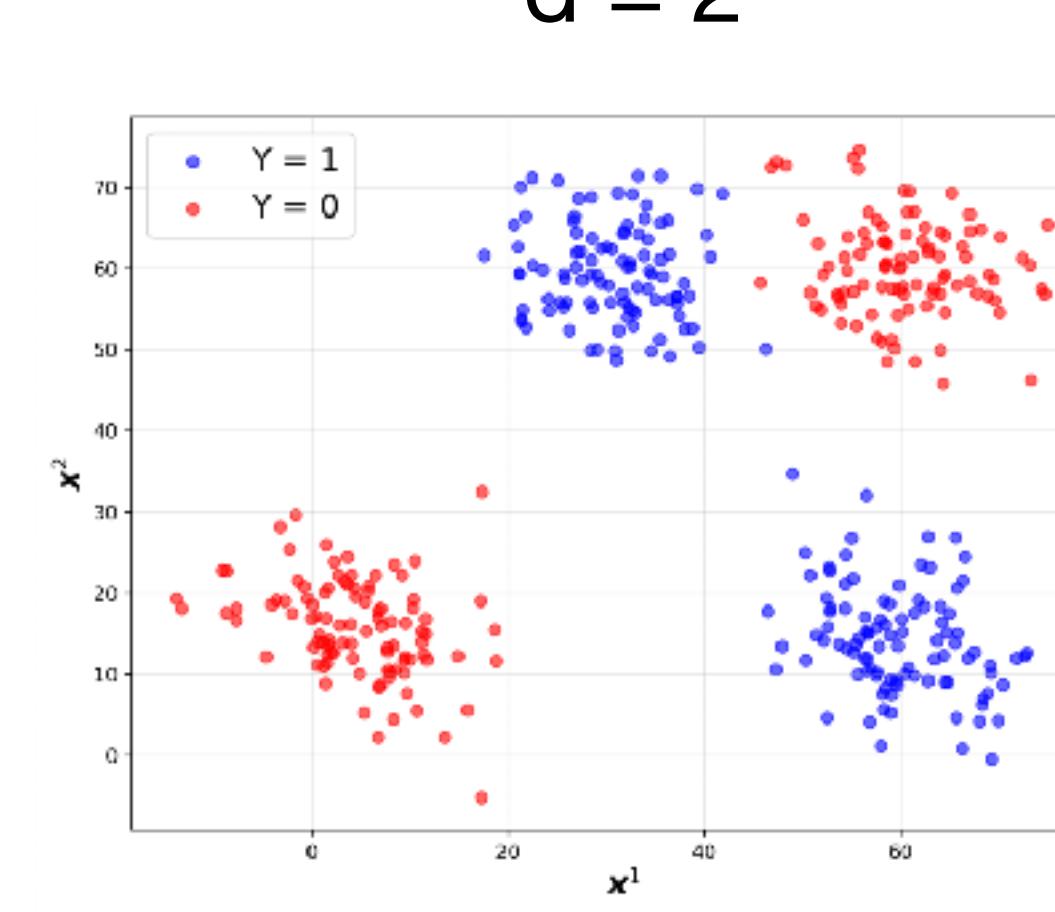
| predictions          | true labels |
|----------------------|-------------|
| $f^*(\mathbf{x}'_1)$ | $y'_1$      |
| $\vdots$             | $\vdots$    |
| $f^*(\mathbf{x}'_m)$ | $y'_m$      |

→ “Test” error

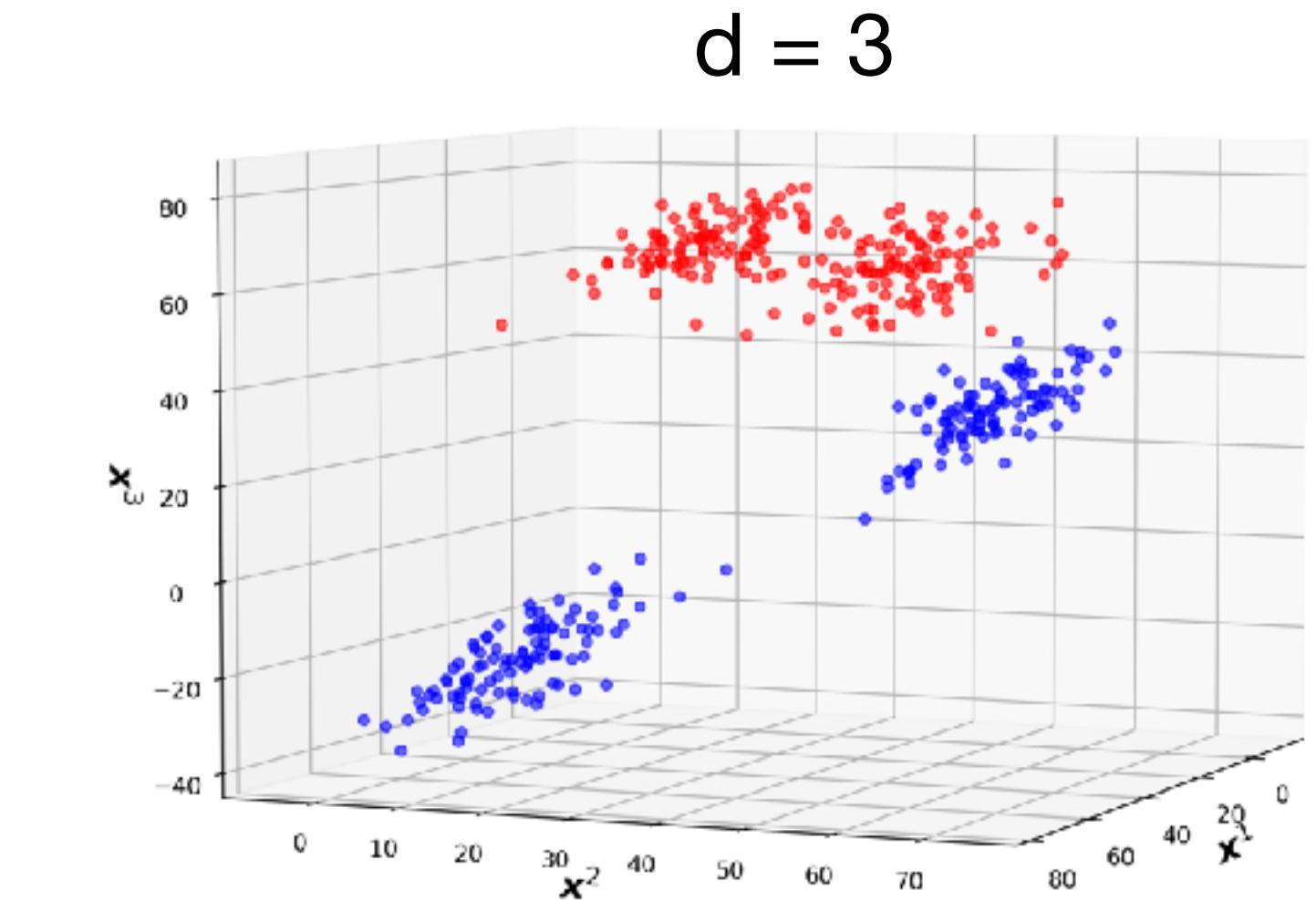
Peut-on séparer les classes avec une séparation linéaire dans ces cas ?



Non !



Non !



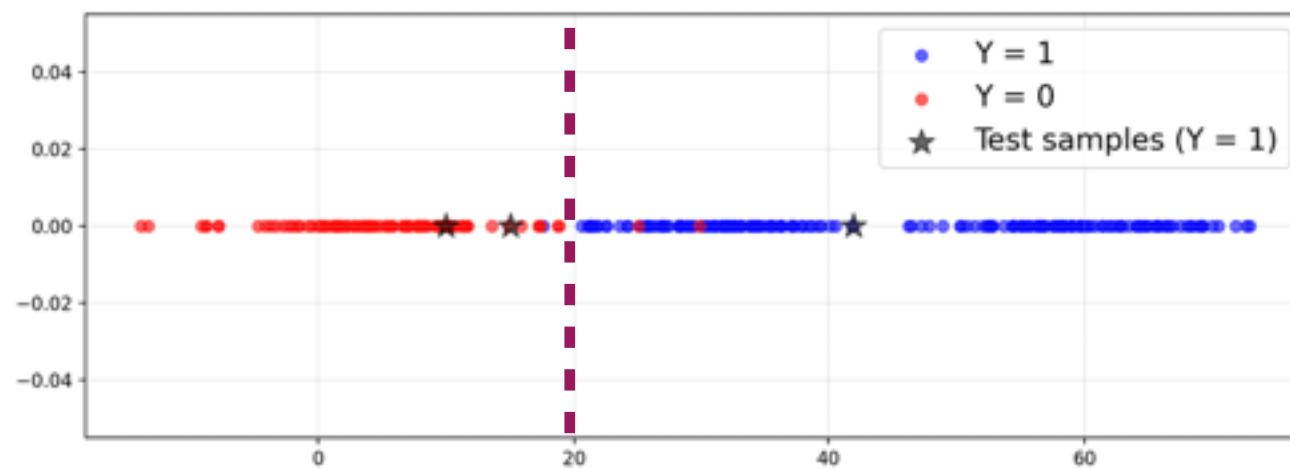
Oui !

$d + 1$  représente le nombre de paramètres à estimer: plus  $d$  est grand, plus le modèle est riche, complexe.

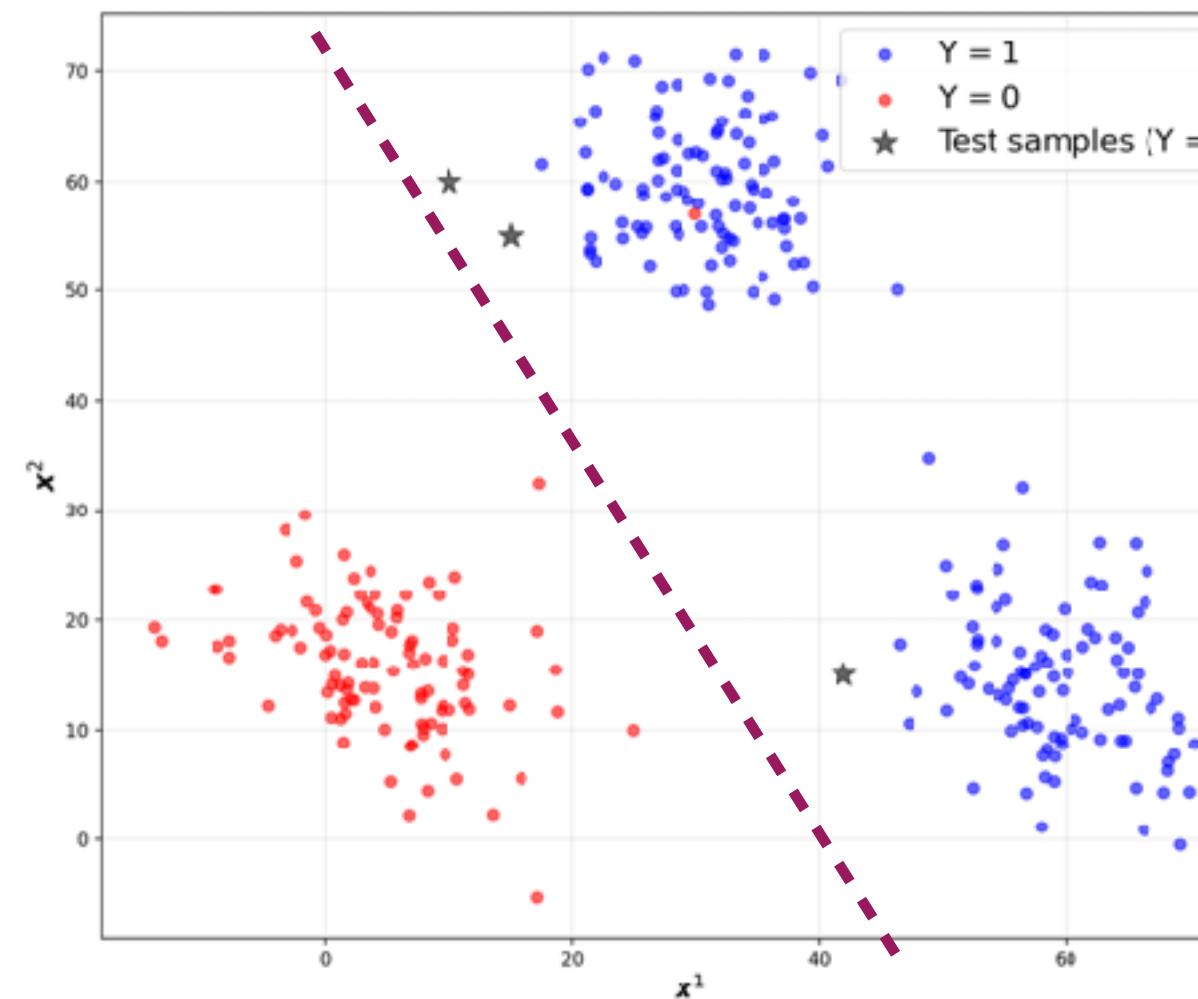
Comment évolue l'erreur sur le train au fur-et-à mesure que la dimension  $d$  augmente ?

Quelle est la meilleure séparation linéaire sur ces données ?

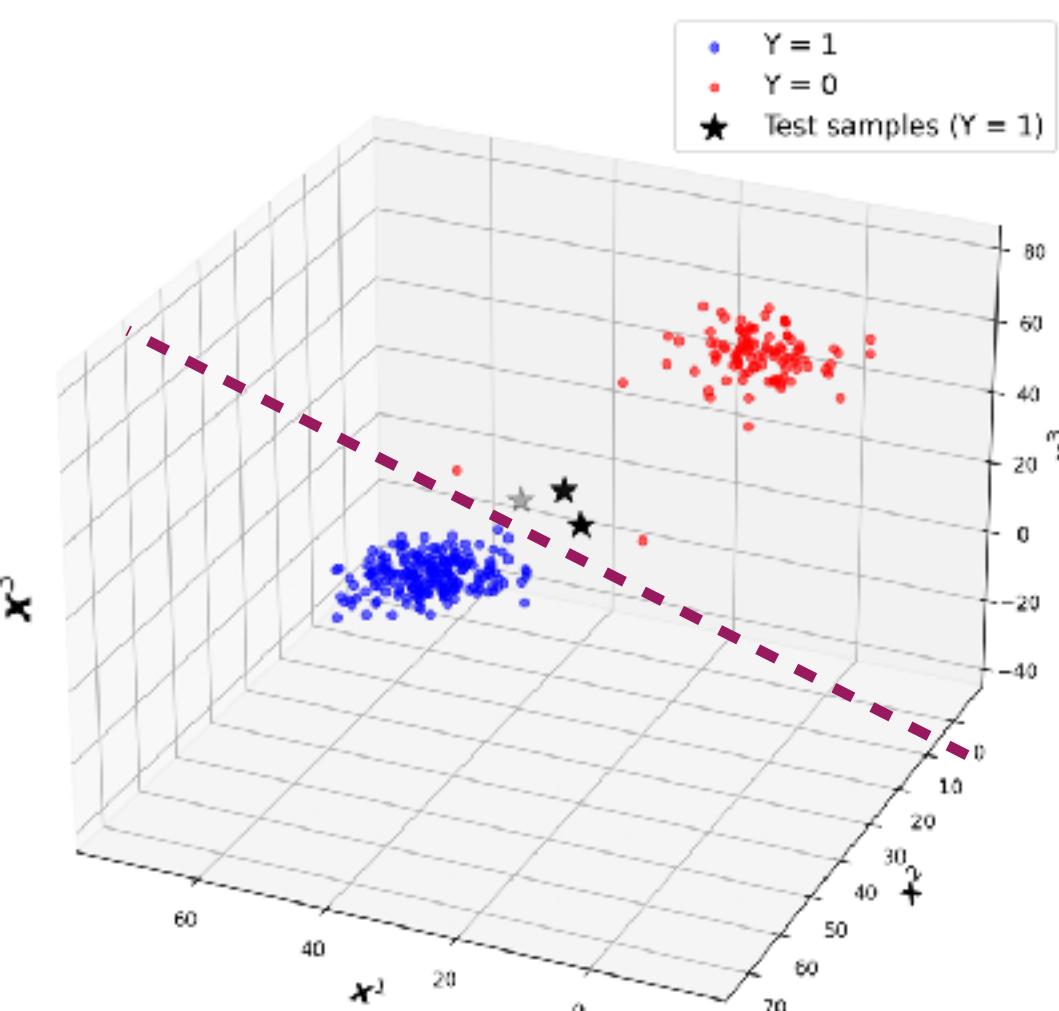
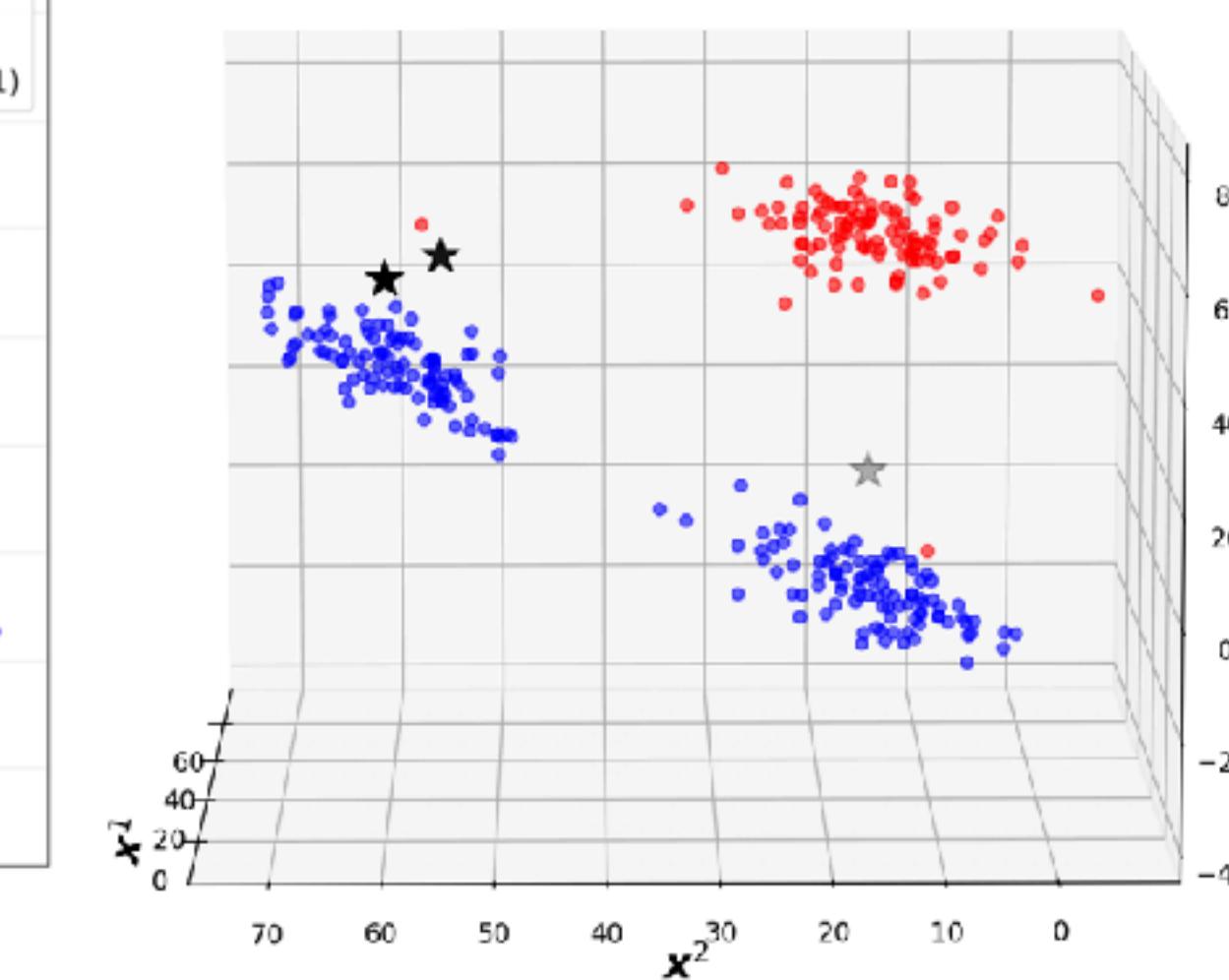
$d = 1$



$d = 2$



$d = 3$



Calculer l'erreur de train et de test.

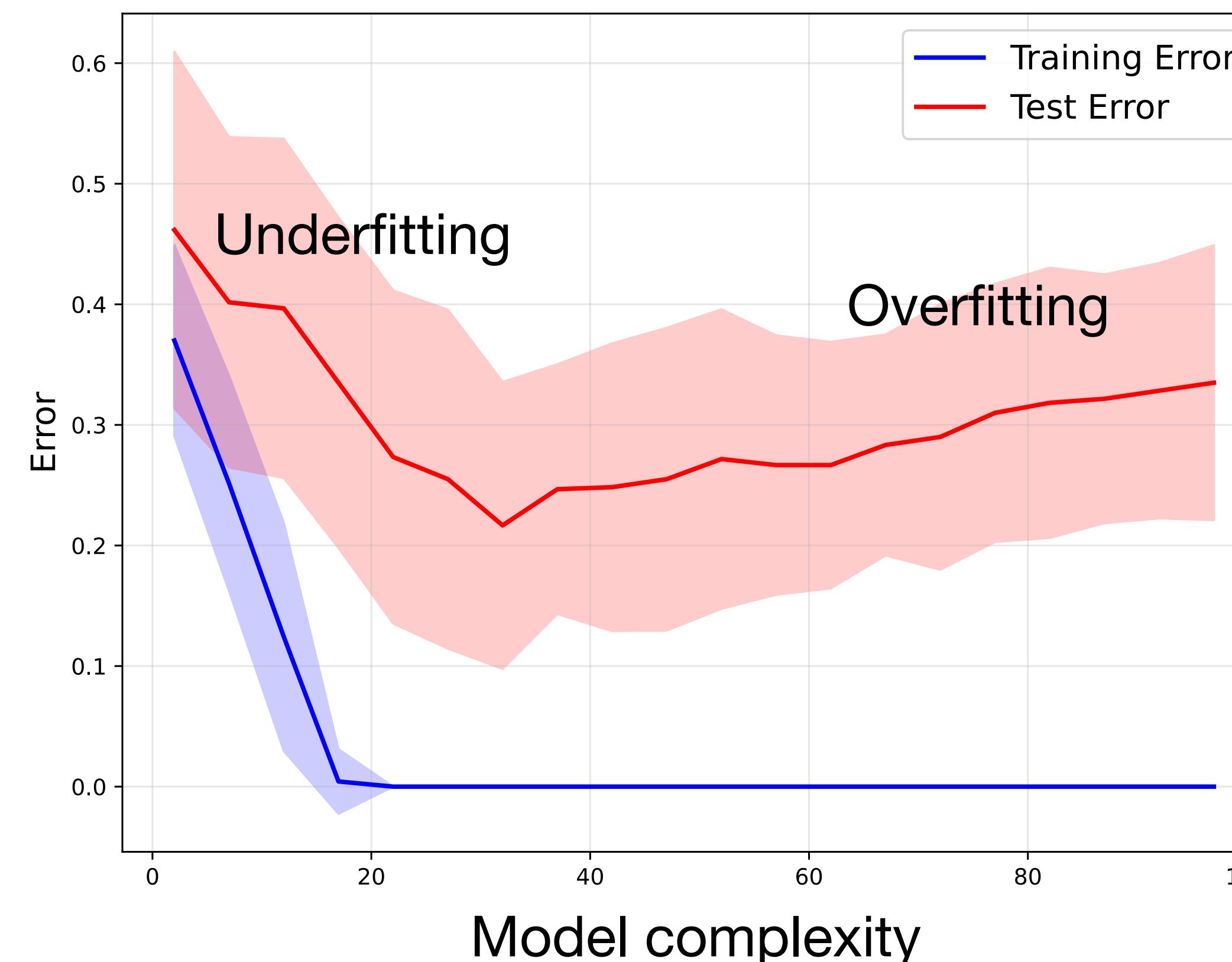
$d = 3$  donne la meilleure erreur de train = 0

$d = 2$  donne la meilleure erreur de test = 0

“La meilleure” séparation linéaire sur le train n'est pas la meilleure sur le test: elle est biaisée par les outliers

Une grande dimension peut causer l'**overfitting**

## “Bias-Variance” tradeoff



Underfitting correspond à:

Grand biais ou grande variance ?

Variance nulle = prédiction constante  
= underfitting

Pour réduire l'overfitting, on peut réduire l'espace d'optimisation en privilégiant des coefficients simples:

$$\min_{\substack{\beta \in \mathbb{R}^{d+1} \\ \|\beta\|_2^2 \leq C}} - \sum_{i=1}^n y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

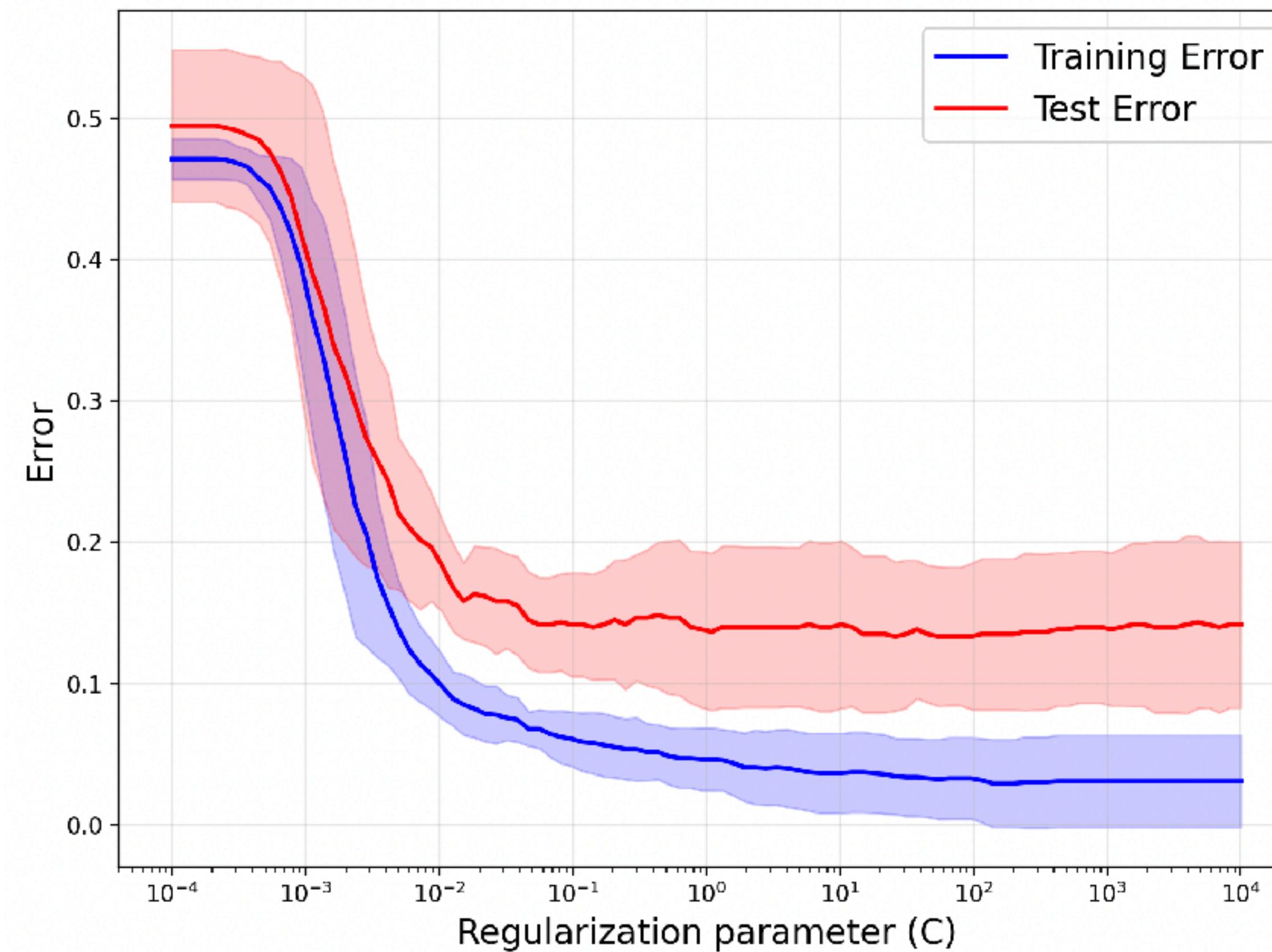
Ce problème n'est pas facile à résoudre (contrainte quadratique), on peut montrer que ce problème est équivalent:

$$\min_{\beta \in \mathbb{R}^{d+1}} - \sum_{i=1}^n y_i \log(p_i) + (1 - y_i) \log(1 - p_i) + \frac{1}{C} \|\beta\|_2^2$$

Dans les deux cas plus  $C$  est petit plus on minimise  $\|\beta\|$ : “Plus on régularise”.

$C \rightarrow 0$  ? le  $\beta$  optimal est le vecteur nul: la fonction de prédiction est constante: underfitting

$C \rightarrow +\infty$  ? l'optimisation est sur  $\mathbb{R}^{d+1}$  en entier: risque d'overfitting.



Tout modèle de machine learning (supervisé) cherche une fonction de prédiction  $f$ .

Supposons qu'elle est paramétrée par  $\theta \in \mathbb{R}^p$ .

Tout modèle de machine learning cherche un compromis entre:

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \text{loss}(f_\theta(\mathbf{x}_i), \mathbf{y}_i) + \frac{1}{C} \text{pénalité}(\theta)$$

Minimiser l'erreur de prédiction  
sur les données "train"

des paramètres "simples" pour  
généraliser à des données  
nouvelles test (éviter l'overfitting)

$C$  contrôle la complexité du modèle

La fonction "pénalité" est aussi appelée "régularisation": elle vient simplifier (régulariser) la fonction de prédiction

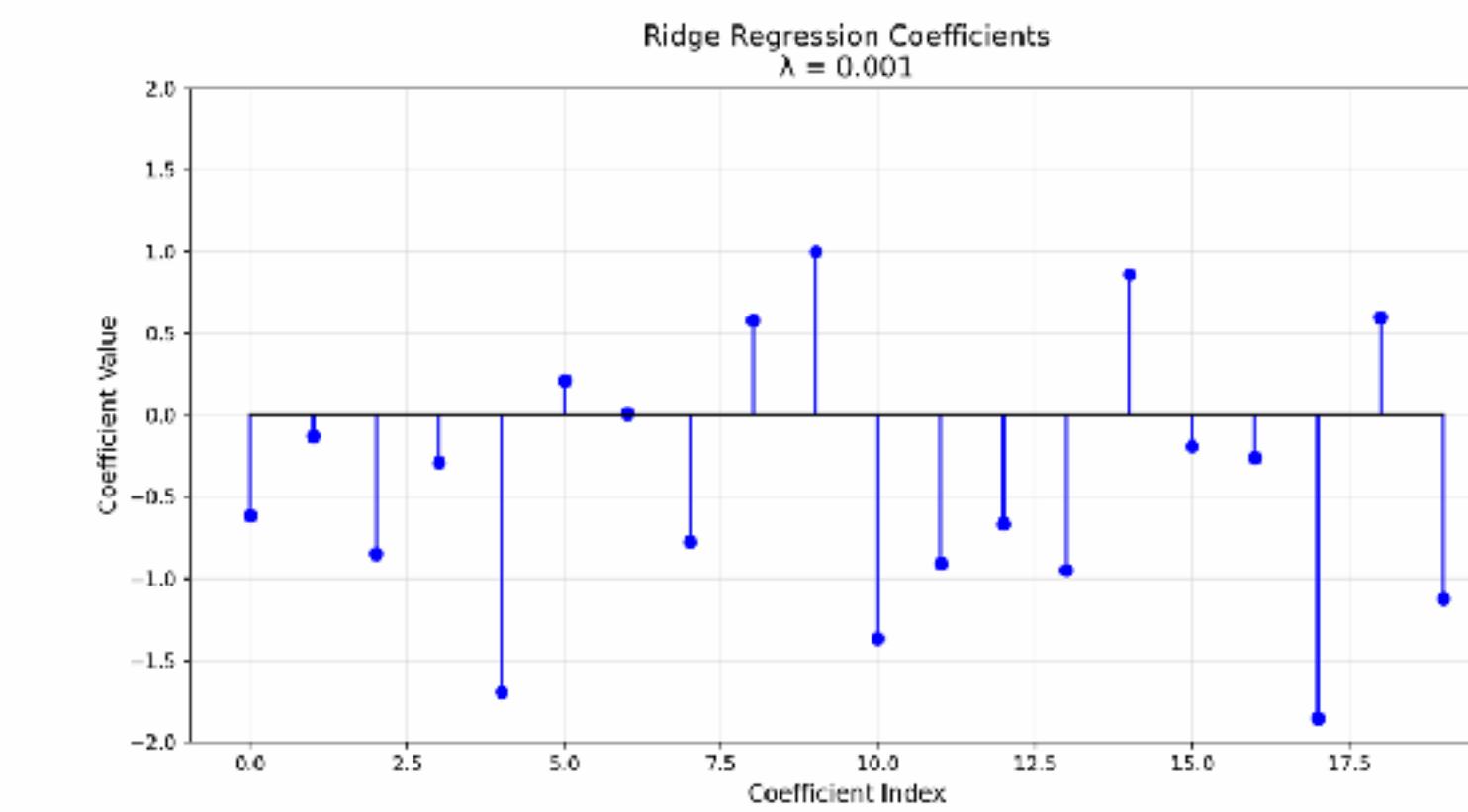
Ce type de régularisation (+ pénalité) est dit: régularisation de Tikhonov

## Comment choisir la pénalité ?

Les pénalités les plus utilisées sont:

1. pénalité Ridge / $\ell_2$  :  $\|\theta\|_2^2$

- 1. Facile à optimiser (différentiable)
- 2. Toutes les variables contribuent au modèle
- 3. Peut être ajoutée à n'importe quel modèle



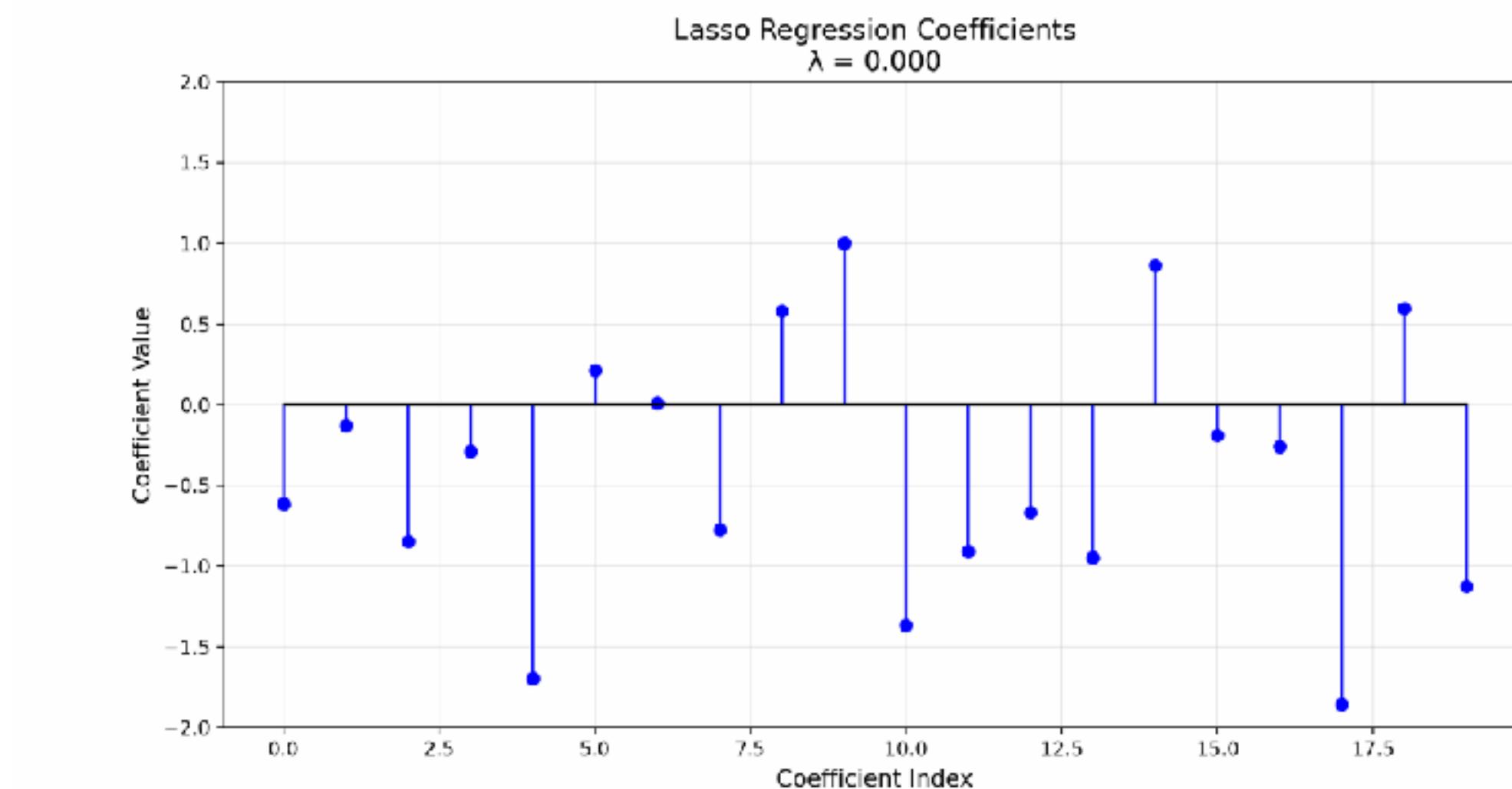
$$\lambda = \frac{1}{C}$$

2. pénalité Lasso / $\ell_1$  :  $\|\theta\|_1$

- 1. Moins facile à optimiser (non-différentiable)
- 2. Permet d'avoir des coefficients "sparses" (beaucoup de 0): utile pour la sélection de variables pertinentes

3. pénalité Elastic net :  $\delta\|\theta\|_2^2 + (1 - \delta)\|\theta\|_1$

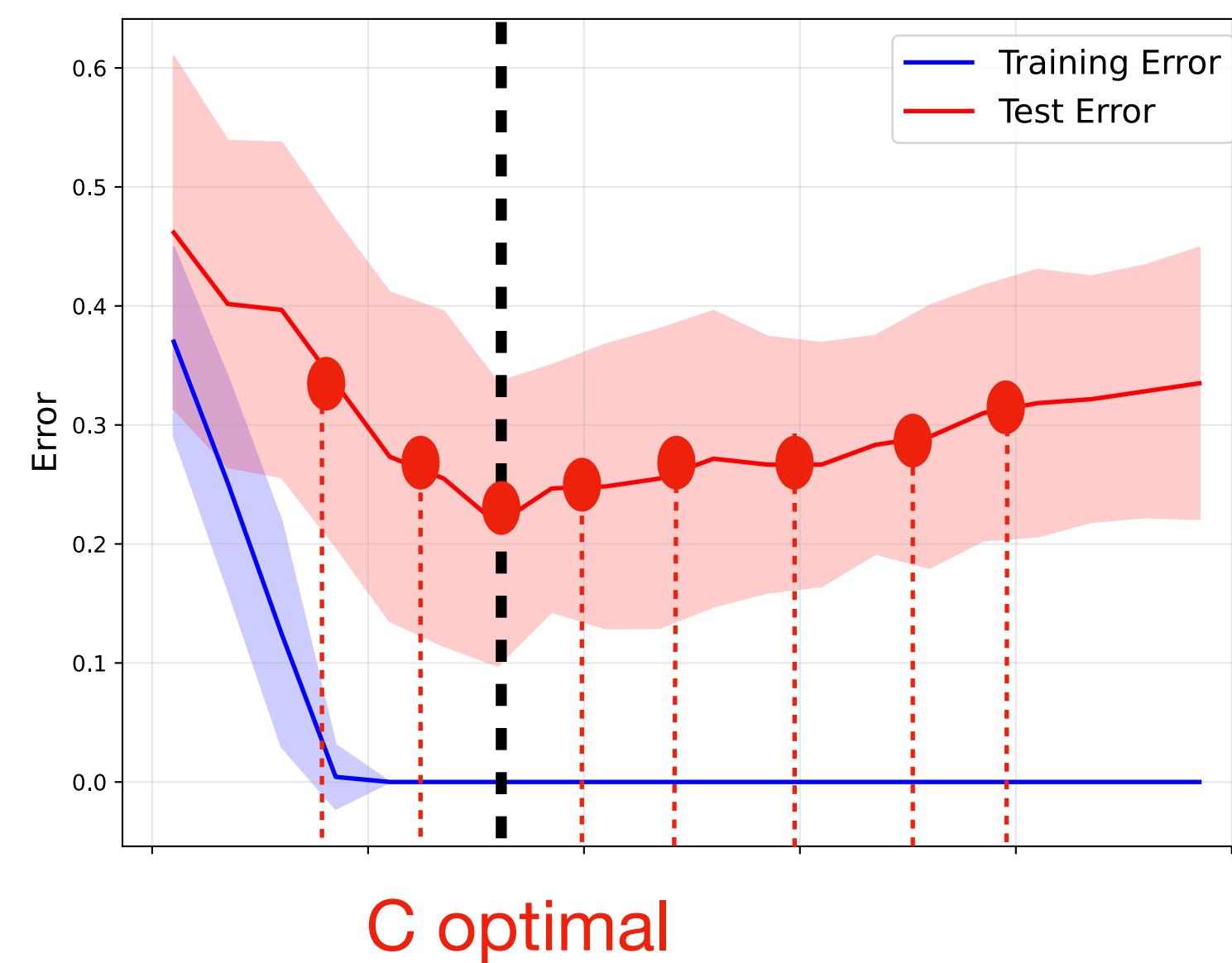
Ridge: "shrink" toutes les coordonnées lentement vers zéro (sans l'atteindre)



Lasso: annule les coefficients un par un

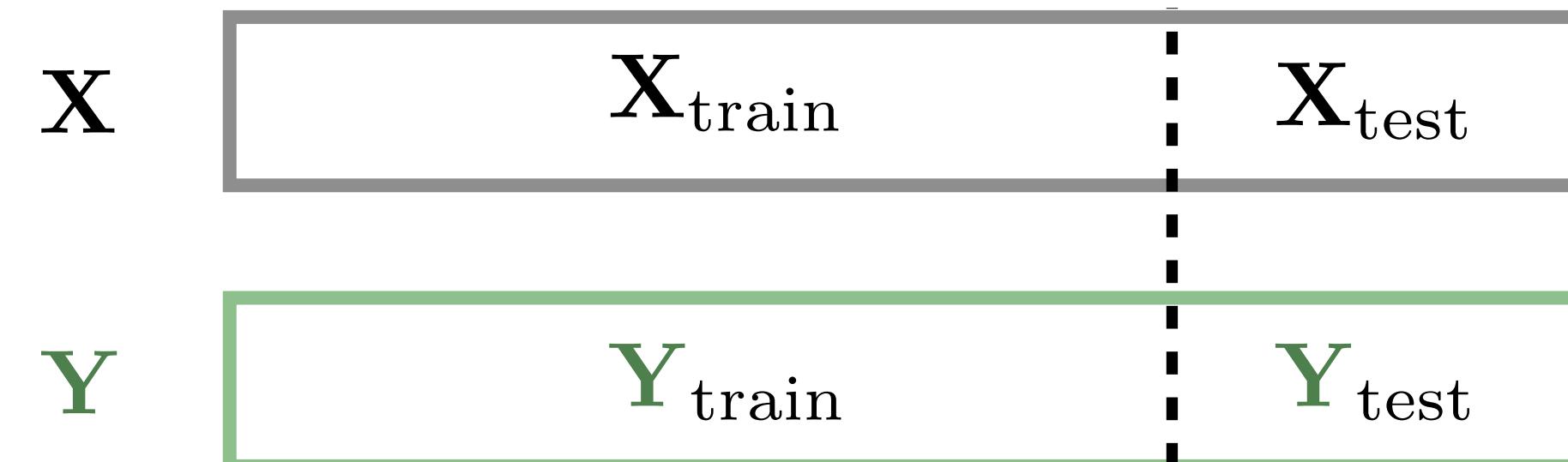
Comment choisir  $C$  ?

On veut le  $C$  qui donne la meilleure performance sur le test



Pour cela on peut:

1. Couper le dataset en deux train et test:



2. Choisir une liste de valeurs de  $C$ , par ex: [0.01, 0.05, 0.1, 1., 10]

Pour chaque  $C$ :

1. Optimiser sur  $\mathbf{X}_{\text{train}}$   $\mathbf{Y}_{\text{train}}$

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \text{loss}(f_{\theta}(\mathbf{x}_i), y_i) + \frac{1}{C} \text{pénalité}(\theta)$$

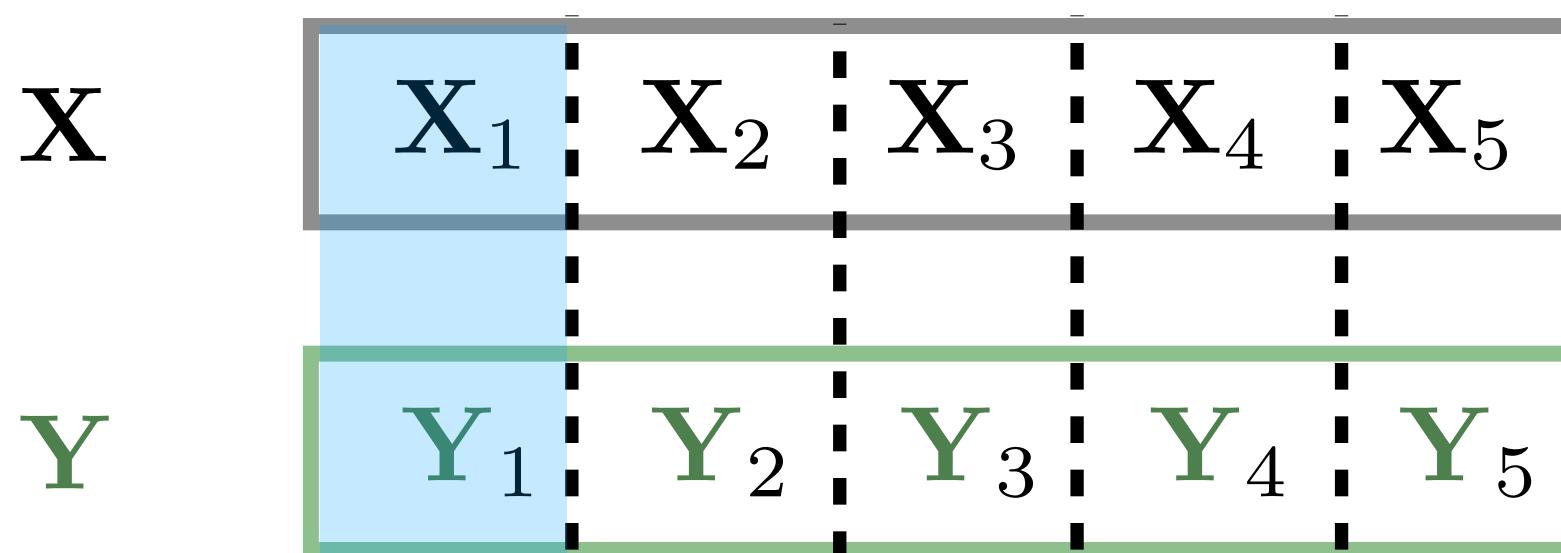
2. Évaluer l'erreur de prédiction sur  $\mathbf{X}_{\text{test}}$   $\mathbf{Y}_{\text{test}}$

3. Choisir la valeur de  $C$  avec la plus petite erreur de prédiction sur le test

Quel est l'inconvénient principal de cette méthode ? Le  $C$  choisi dépend du découpage aléatoire train / test

Idée: Effectuer plusieurs découpages et moyenner l'erreur de test

1. Couper le dataset en 5 parties (folds)



2. Choisir une liste de valeurs de  $C$ , par ex: [0.01, 0.05, 0.1, 1., 10]

3. Pour chaque  $k$  in [1, 2, 3, 4, 5], créer un découpage train/test

$$\mathbf{X}_{\text{test}}, \mathbf{Y}_{\text{test}} = \mathbf{X}_k, \mathbf{Y}_k$$

$$\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}} = [\mathbf{X} \text{ sans } \mathbf{X}_k], \dots, [\mathbf{Y} \text{ sans } \mathbf{Y}_k]$$

Pour chaque  $C$ :

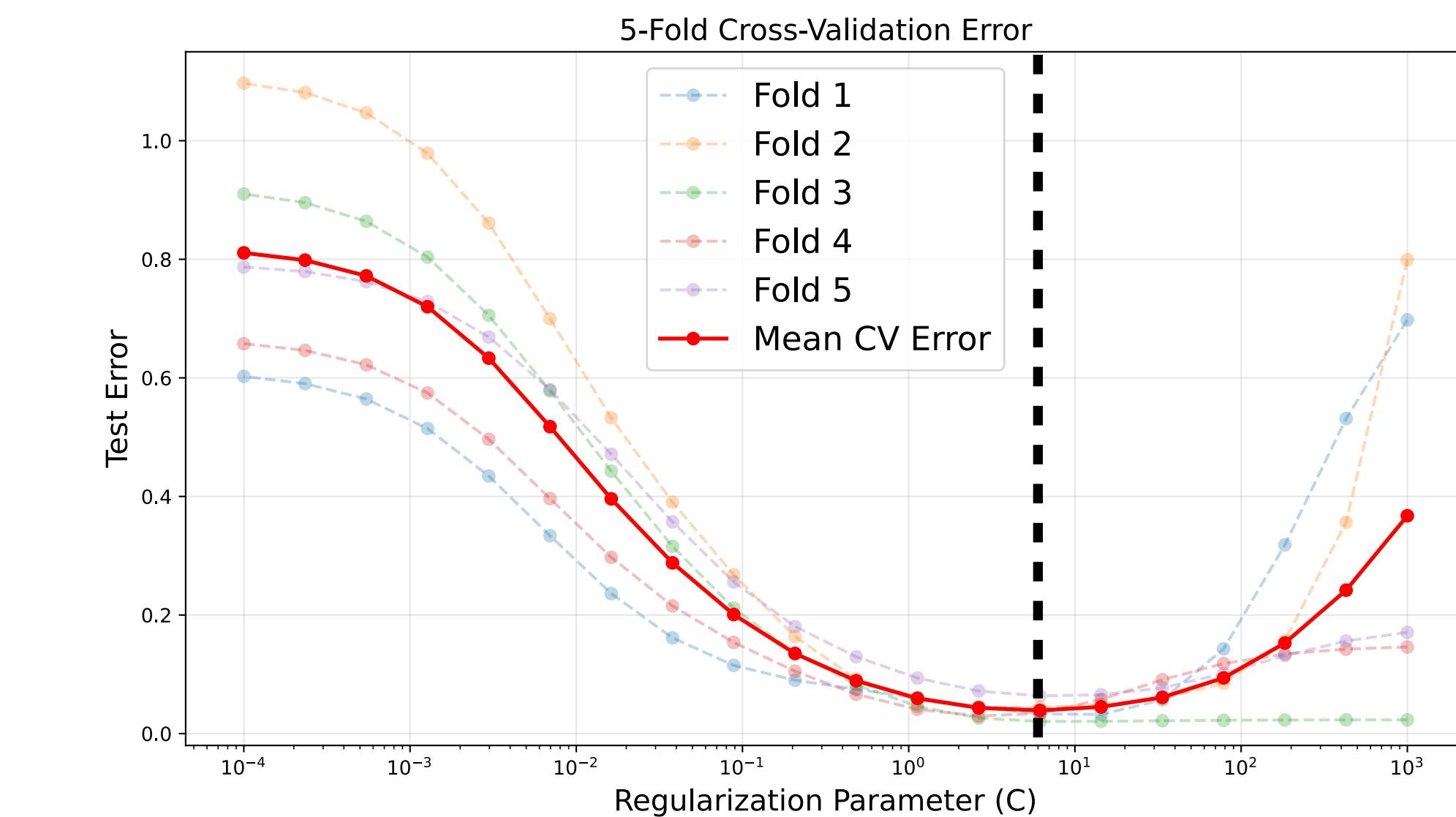
1. Optimiser sur

$$\mathbf{X}_{\text{train}} \quad \mathbf{Y}_{\text{train}} \quad \min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \text{loss}(f_{\theta}(\mathbf{x}_i), y_i) + \frac{1}{C} \text{pénalité}(\theta)$$

2. Évaluer l'erreur de prédiction sur

4. Pour chaque  $C$ , calculer l'erreur de prédiction moyenne

5. Choisir le  $C$  avec l'erreur de prédiction moyenne la plus petite



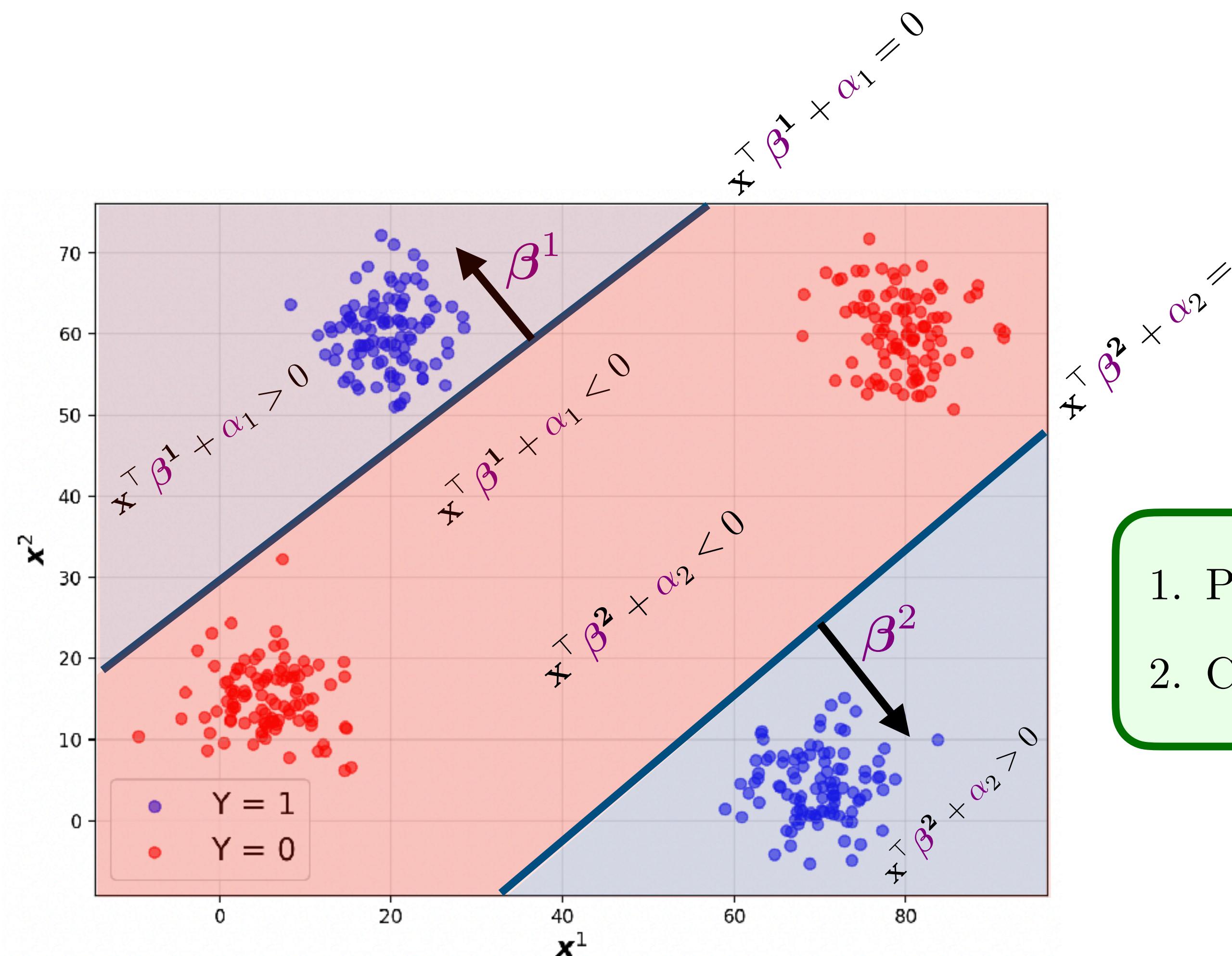
$C$  optimal

C'est l'erreur de validation croisée

5-Fold cross validation



Et si les données ressemblent à ceci ?



Aucune fonction linéaire ne peut séparer les classes

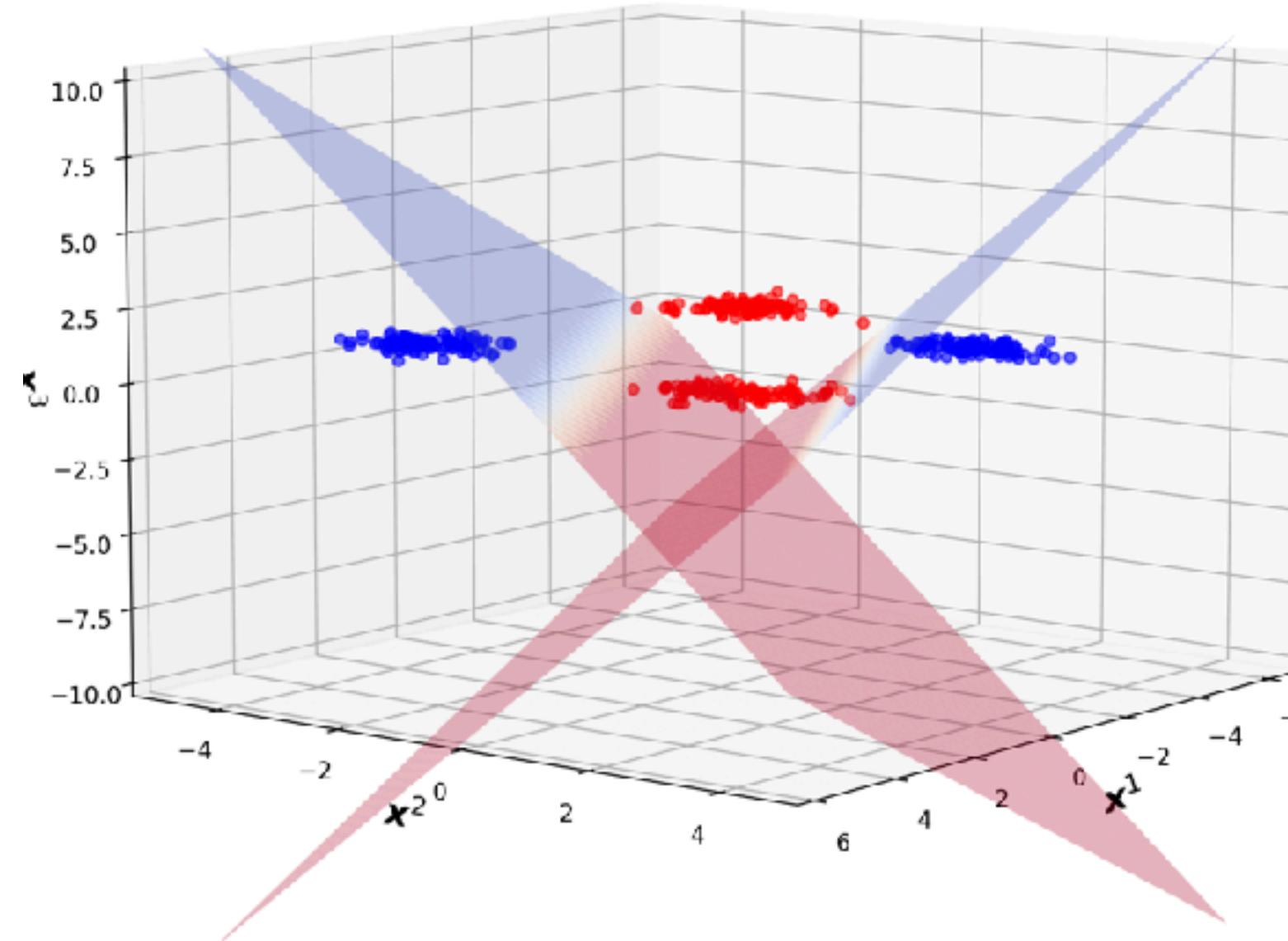
Idée: “combiner” plusieurs fonctions linéaires

$$z_1 \stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta}^1 + \alpha_1$$

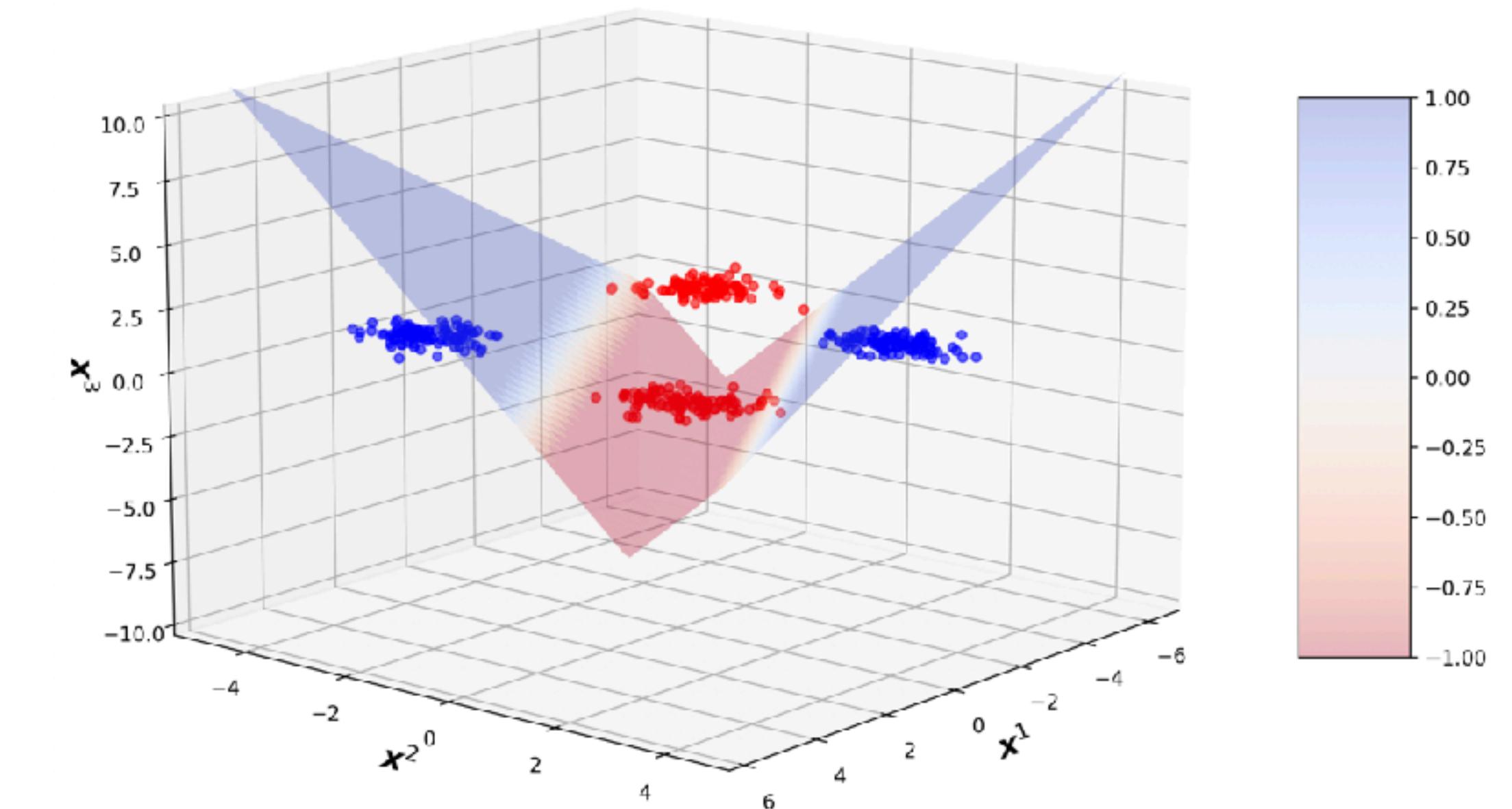
$$z_2 \stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta}^2 + \alpha_2$$

1. Prendre des  $\mathbf{x}$  dans  $\mathbb{R}^2$  et étudier les signes possibles de  $z_1, z_2$ .
2. Comment peut-on prédire  $Y = 1$  à partir des  $z_i$  ?

Surfaces des hyperplans  $z_1, z_2$



Surface de  $\max(z_1, z_2)$



Prédire  $Y = 1$  si l'un des  $z_i$  est positif  $\Leftrightarrow \max(z_1, z_2) > 0$

$$f_{\alpha, \beta}(\mathbf{x}) = \mathbb{1}_{\{\max(\mathbf{x}^\top \boldsymbol{\beta}^1 + \alpha_1, \mathbf{x}^\top \boldsymbol{\beta}^2 + \alpha_2) > 0\}}$$

Linear functions

Comment entraîner ce modèle, c-à-d optimiser  $\alpha, \beta$  ?

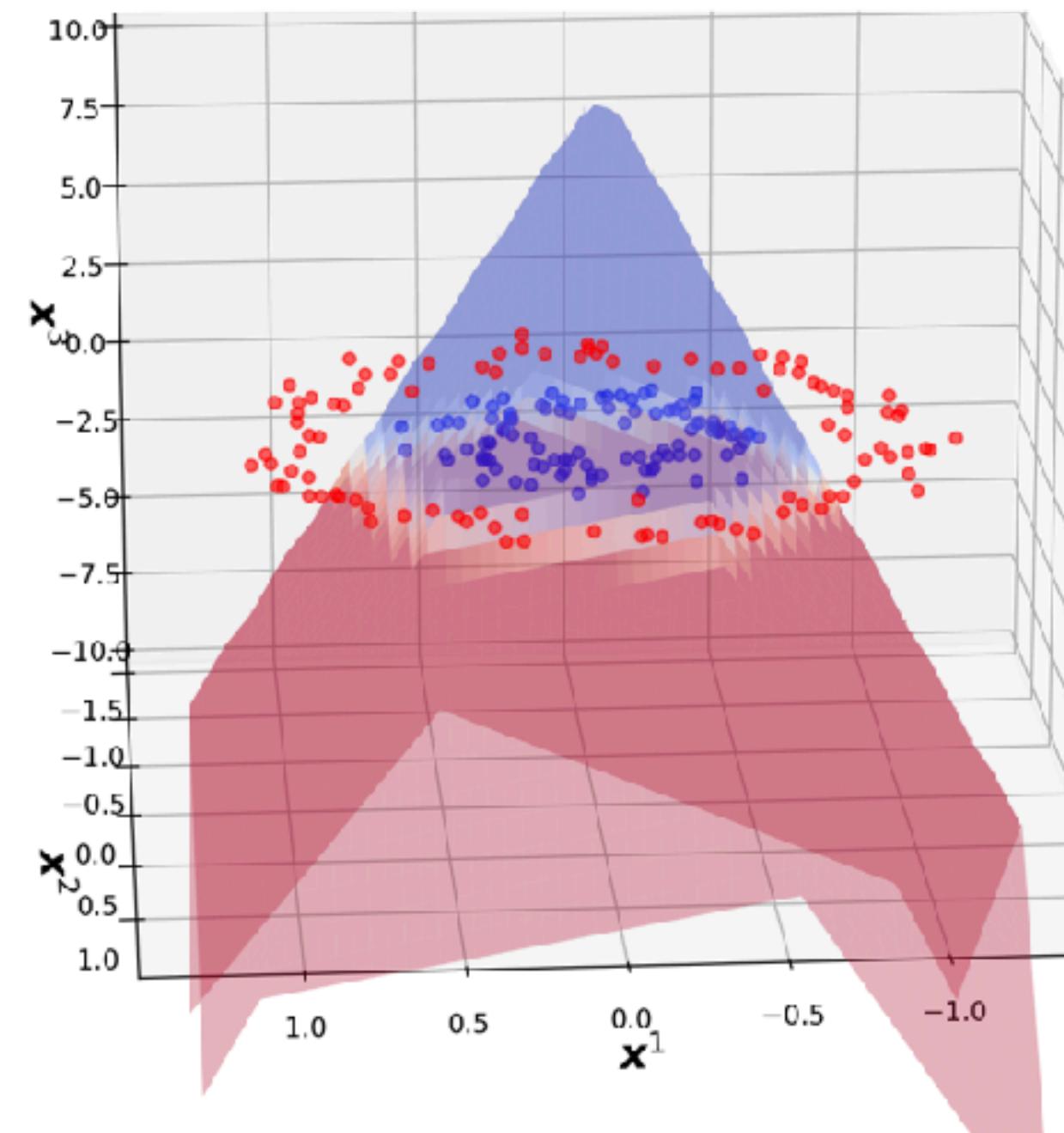
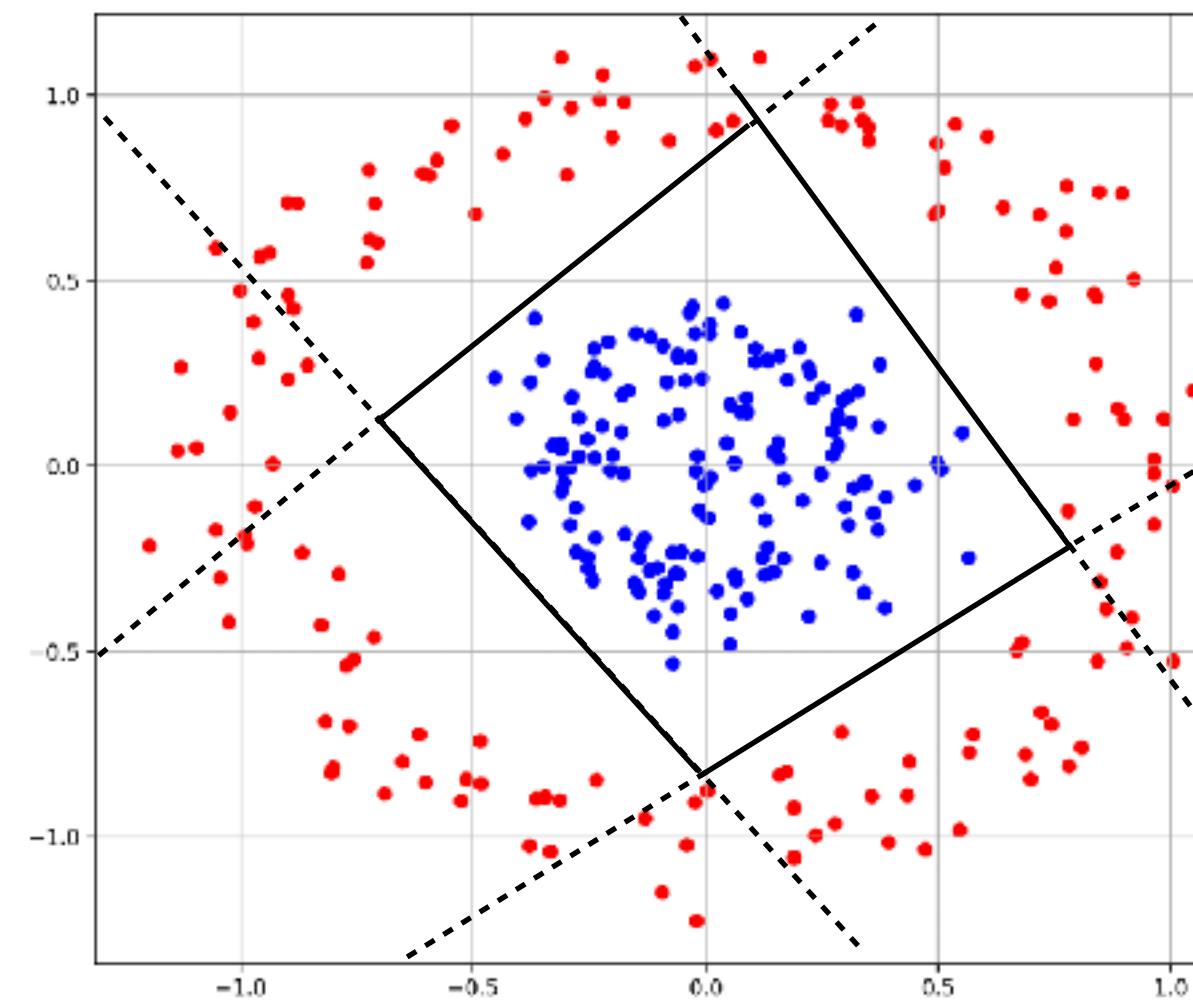
$$p_i \stackrel{\text{def}}{=} \mathbb{P}_{\alpha, \beta}(Y = 1 | \mathbf{x}_i) = \text{sigmoid}(\max(\mathbf{x}_i^\top \boldsymbol{\beta}^1 + \alpha_1, \mathbf{x}_i^\top \boldsymbol{\beta}^2 + \alpha_2))$$

Non-linearity

Comme la régression logistique:

$$\min_{\alpha \in \mathbb{R}, \beta \in \mathbb{R}^d} - \sum_{i=1}^n y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

Comment adapter ce modèle à des données plus complexes ?



Linéarités

$$z_1 \stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta^1} + \alpha_1$$

$$z_2 \stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta^2} + \alpha_2$$

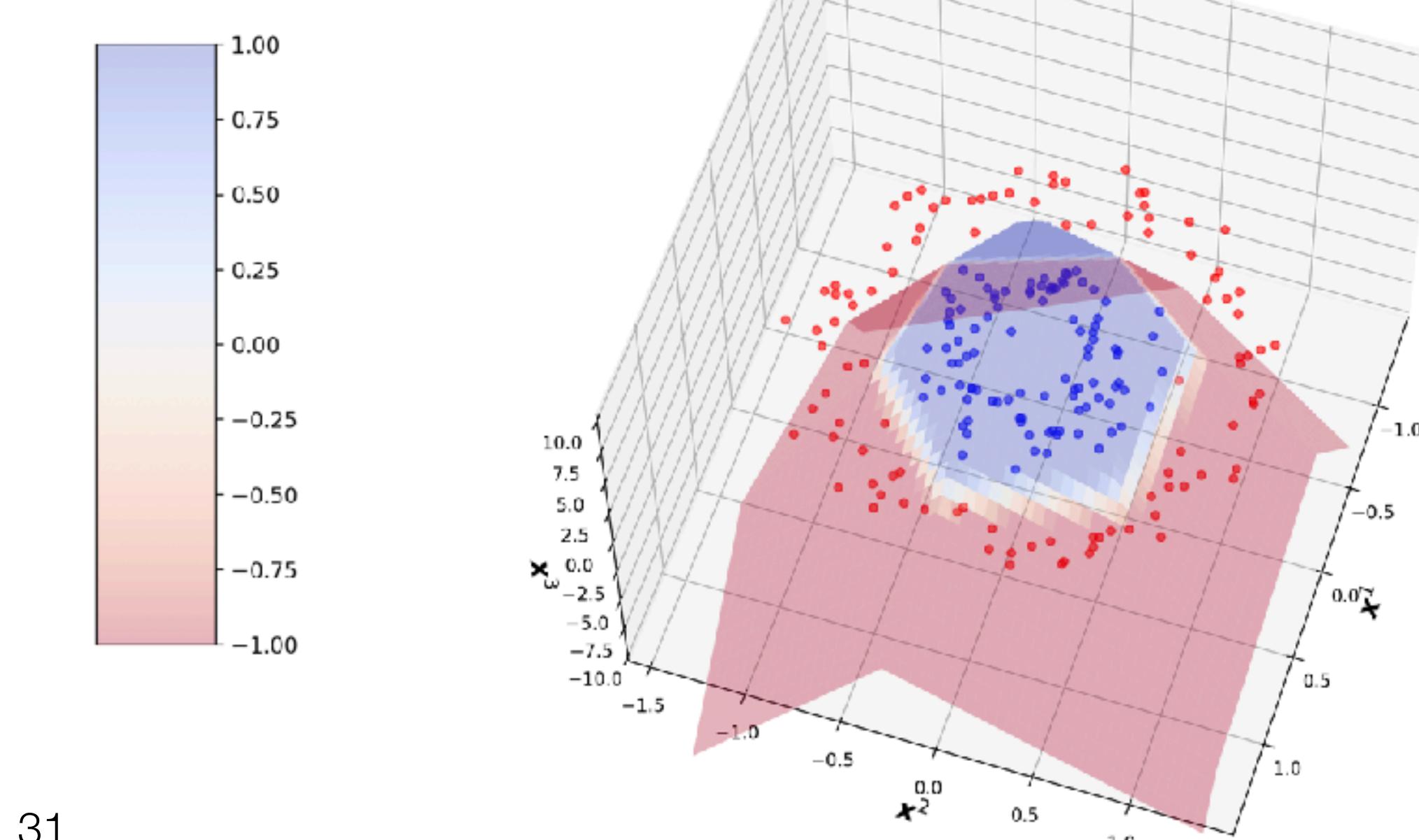
⋮

$$z_p \stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta^p} + \alpha_p$$

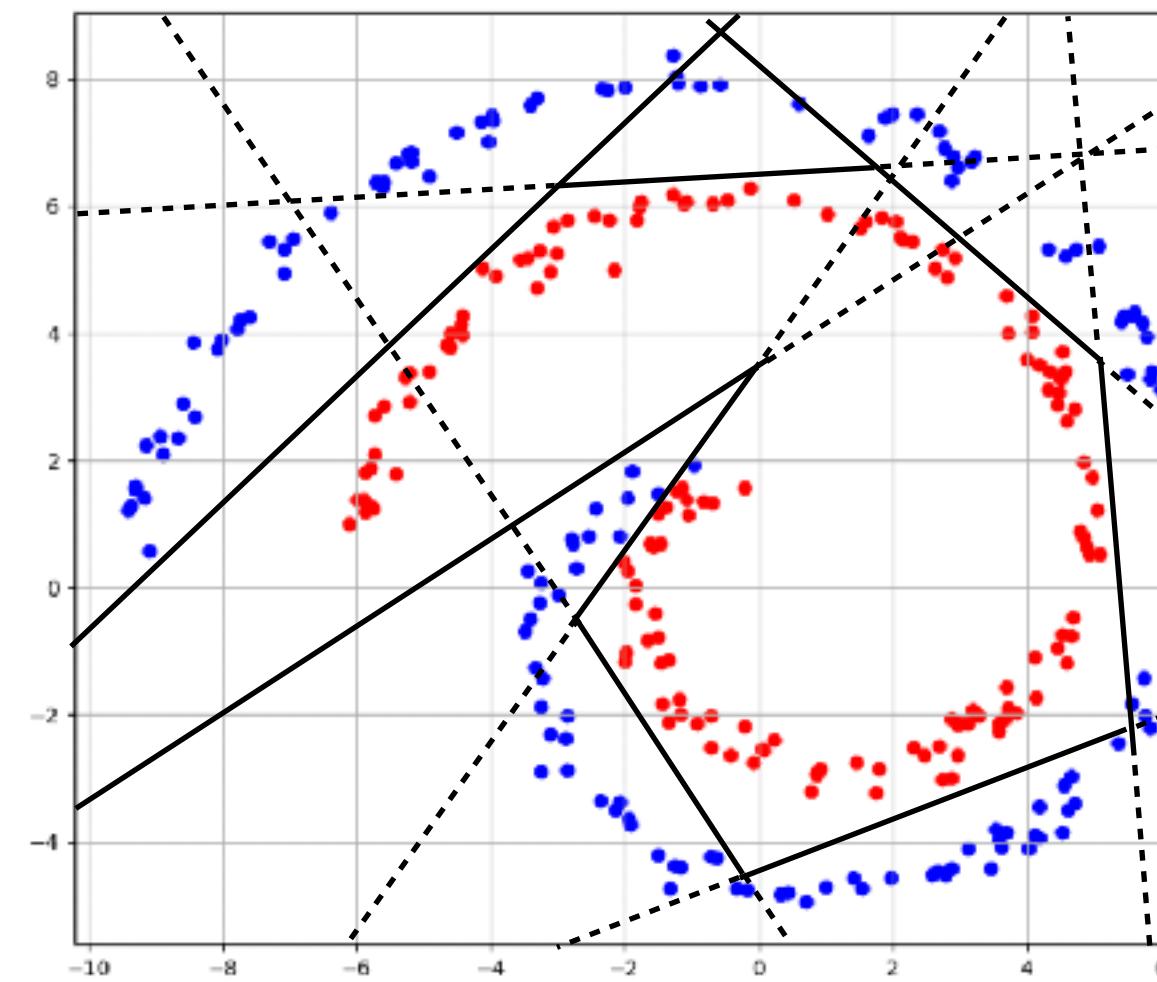
non-linéarité

$$\max(z_1, \dots, z_p)$$

sigmoid



Comment adapter ce modèle à des données plus complexes ?



Linéarités

$$z_1 \stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta^1} + \alpha_1$$

$$z_2 \stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta^2} + \alpha_2$$

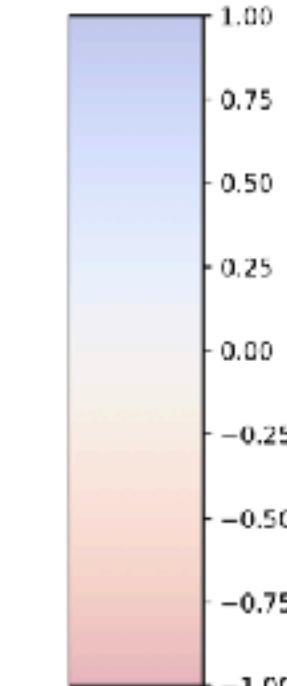
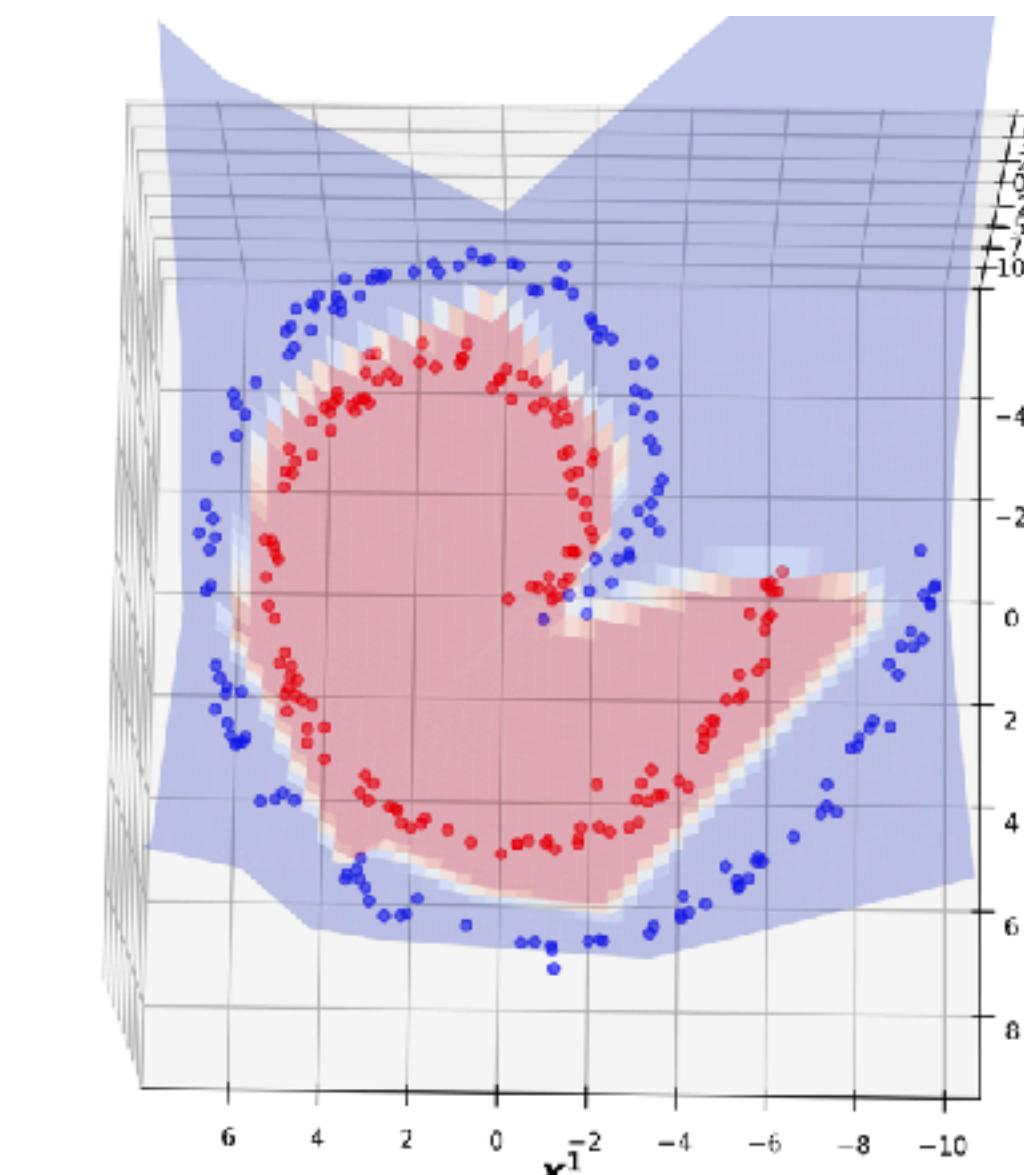
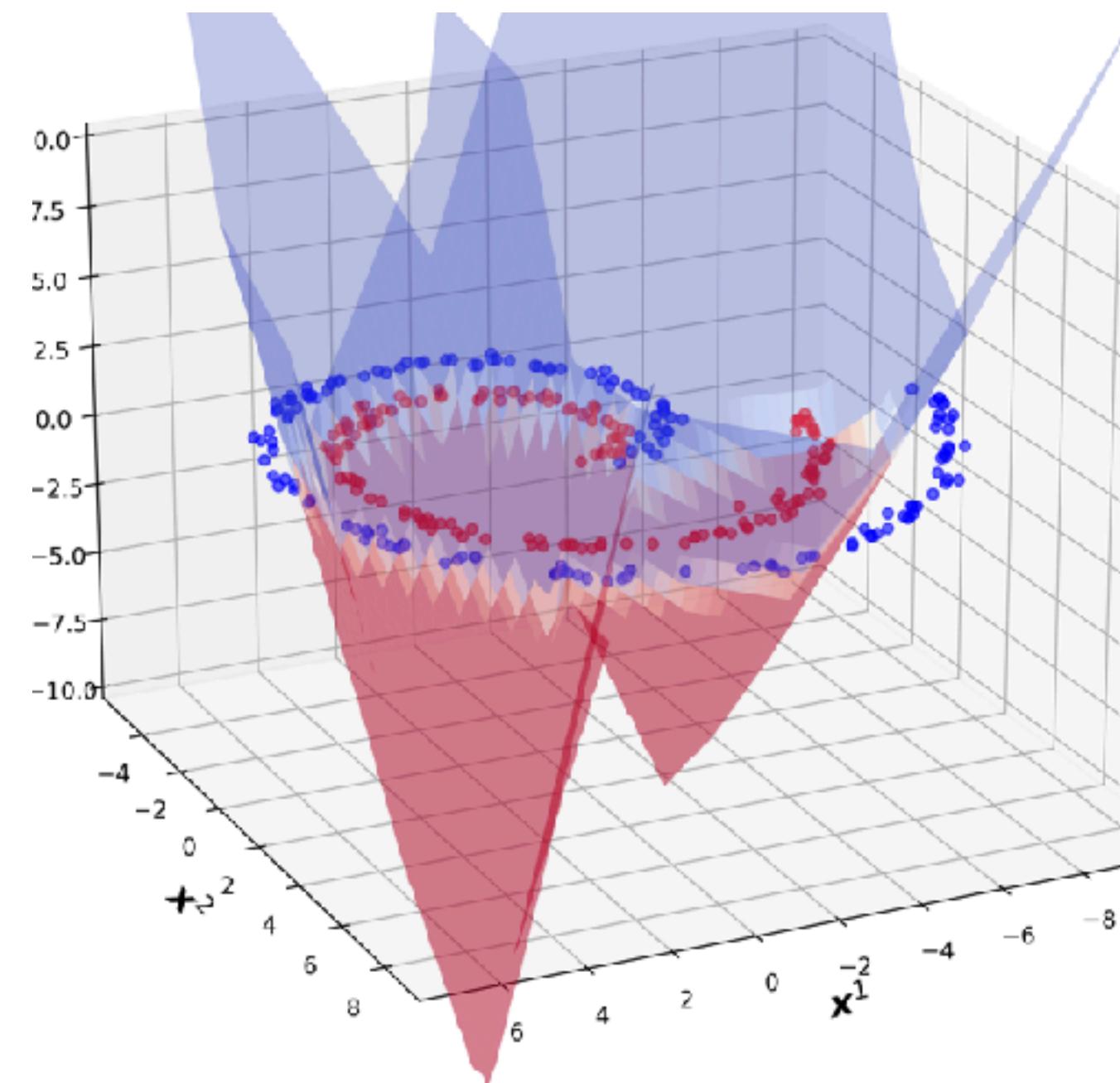
⋮

$$z_p \stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta^p} + \alpha_p$$

non-linéarité

$$\max(z_1, \dots, z_p)$$

sigmoid



Linéarités

$$\begin{aligned} z_1 &\stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta^1} + \alpha_1 \\ &\vdots \\ z_p &\stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta^p} + \alpha_p \end{aligned}$$

non-linéarité

$\max(z_1, \dots, z_p) \longrightarrow$  sigmoid

En pratique, ce modèle ne fonctionne pas pour ces données complexes. Pourquoi à votre avis ?

1. On n'utilise qu'**une seule** non-linéarité
2. Elle est fixée par la fonction **max**: on ne l'apprend pas

Il faudrait donc: utiliser plusieurs **non-linéarités simples** + les combiner pour apprendre des fonctions non-linéaires complexes

Idée:

1. Appliquer plusieurs non-linéarités ***h*** plus tôt
2. Combiner les  $z_j$  linéairement avec  $w_j$  à optimiser

$$\begin{aligned} z_1 &\stackrel{\text{def}}{=} h(\mathbf{x}^\top \boldsymbol{\beta^1} + \alpha_1) \\ &\vdots \\ z_p &\stackrel{\text{def}}{=} h(\mathbf{x}^\top \boldsymbol{\beta^p} + \alpha_p) \end{aligned} \longrightarrow \sum_{j=1}^p \omega_j z_j + \omega_0$$

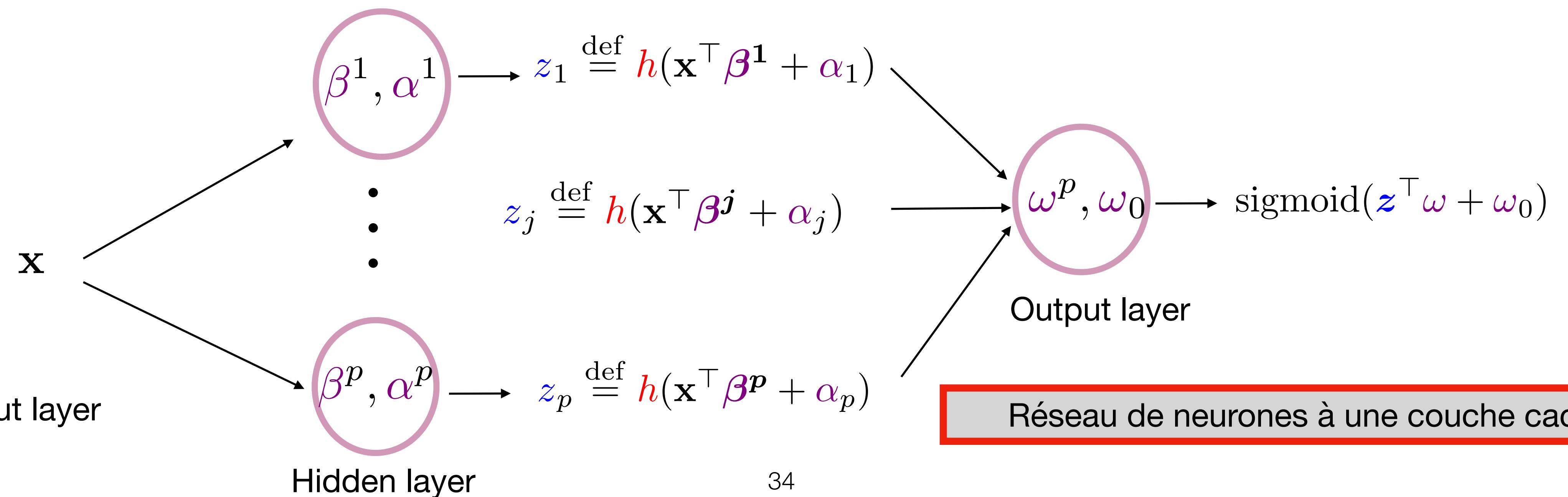
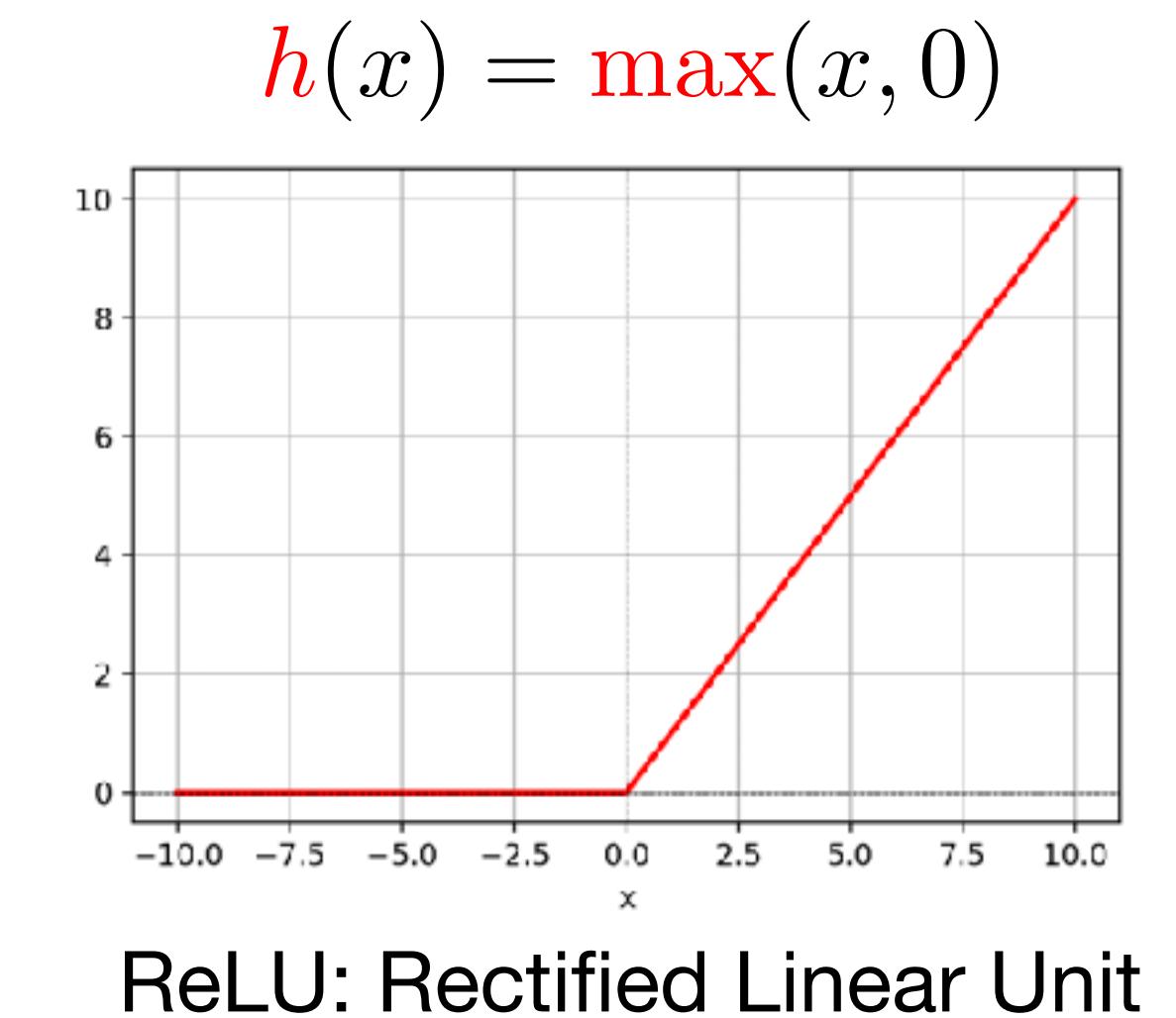
sigmoid



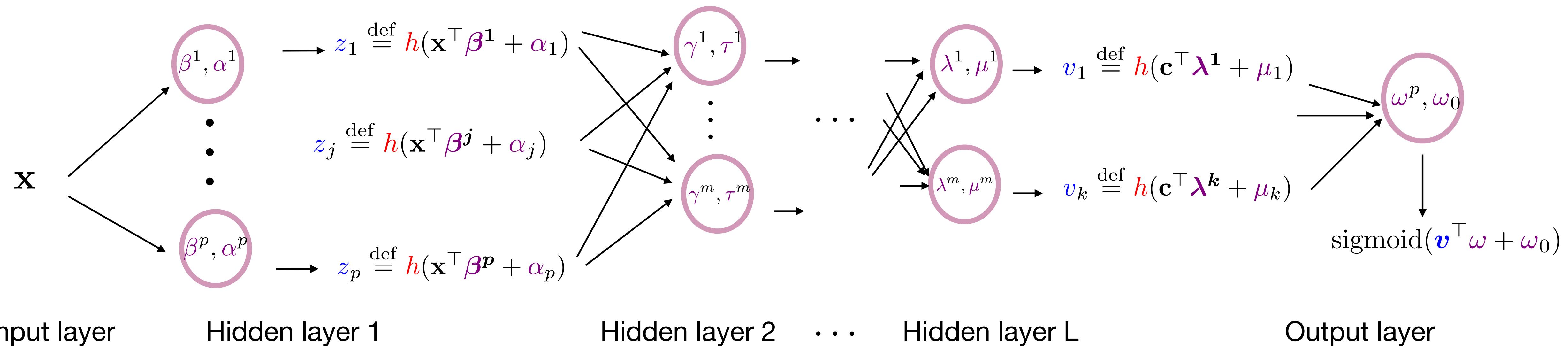
$$\begin{aligned} z_1 &\stackrel{\text{def}}{=} h(\mathbf{x}^\top \boldsymbol{\beta}^1 + \alpha_1) \\ &\vdots \\ z_p &\stackrel{\text{def}}{=} h(\mathbf{x}^\top \boldsymbol{\beta}^p + \alpha_p) \end{aligned} \quad \text{sigmoid}\left(\sum_{j=1}^p \omega_j z_j + \omega_0\right) = \text{sigmoid}(\mathbf{z}^\top \boldsymbol{\omega} + \omega_0)$$

Quelle est la fonction non-linéaire  $h$  la plus simple possible ?

On représente ce type de modèle sous forme de graphe avec des “unités” de calcul simples: fonction linéaire + non-linéarité. Unité = un neurone:



On peut augmenter la complexité du modèle à l'infini...



Deep neural networks = many layers / many neurons

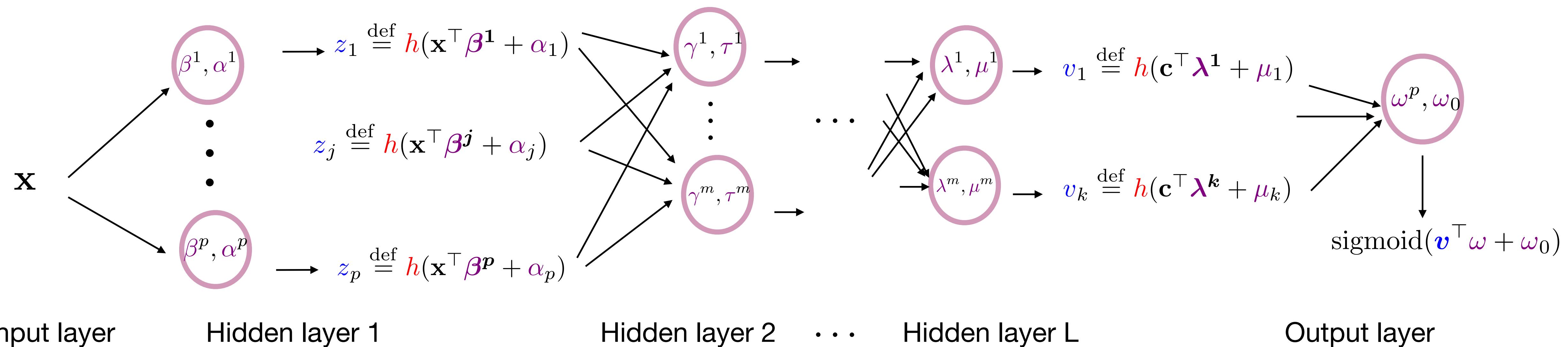
This is a “general purpose” neural network (NN) known as “fully connected multilayer perceptron” (MLP)

On considère un problème de classification binaire avec le réseau ci-dessus optimisé avec une très bonne performance.

On définit la transformation des données en s’arrêtant à l’avant dernier layer:  $g : \mathbf{x} \in \mathbb{R}^d \mapsto \mathbf{v} \in \mathbb{R}^k$

Apprendre à classifier les  $g(\mathbf{x}_i)$  est-il plus facile ou plus difficile que classifier les  $\mathbf{x}_i$  ?

On peut augmenter la complexité du modèle à l'infini...



On considère un problème de classification binaire avec le réseau ci-dessus optimisé avec une très bonne performance.

On définit la transformation des données en s'arrêtant à l'avant dernier layer:  $g : \mathbf{x} \in \mathbb{R}^d \mapsto \mathbf{v} \in \mathbb{R}^k$

Apprendre à classifier les  $g(\mathbf{x}_i)$  est-il plus facile ou plus difficile que classifier les  $\mathbf{x}_i$  ?

Plus **facile**: car un seul neurone (output) a suffit pour les classifier: ils sont **forcément** linéairement séparables

$g(\mathbf{x}_i)$  est un *embedding* ou une *représentation vectorielle* de  $\mathbf{x}_i$

# Chapitre 3. Applications et thématiques avancées

1. Modèles Bayésiens hiérarchiques (Assurance / Biostats)
2. Classical Machine learning: zero to hero
3. Bayesian Machine learning

**Idée principale:** Les paramètres optimisés  $\theta \in \mathbb{R}^p$  en ML deviennent des variables aléatoires suivant une loi a priori  $\pi$  dont la variance est donnée par  $C$ .

Discuter de l'effet de  $C$  très grand / très petit.

$C \rightarrow \infty$ : aucun a priori, toutes les valeurs sont possibles:  $\Rightarrow$  risque d'overfitting

$C \rightarrow 0$ : a priori très strict, seule une valeur est possible  $\Rightarrow$  risque d'underfitting

La loi a priori est équivalente à la pénalité de régularisation: elle contrôle la complexité du modèle

**Différence de méthodologie:** Au lieu d'optimiser, on simule suivant la loi a posteriori  $\theta | \text{data}$

**Résultats:** Au lieu d'avoir une seule prédiction on a une **distribution** de prédictions: moyenne, écart-type, HDI ...

## Frequentist machine learning

Fonction de prédiction  $f_{\theta}$  paramétrée par  $\theta \in \mathbb{R}^p$ .

On optimise:

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \text{loss}(f_{\theta}(\mathbf{x}_i), y_i) + \frac{1}{C} \text{pénalité}(\theta)$$

On obtient une fonction de prédiction optimale  $f_{\theta^*}$

Choisir le meilleur  $C$  par validation croisée

1. Facile à expliquer et à implémenter
2. Adapté pour des quantités de données gigantesques (Optimisation distribuée, stochastique)
3. Optimisation (souvent) non-convexe: dépend de l'initialisation

## Bayesian machine learning

$\theta$  est un vecteur aléatoire suivant une loi a priori  $\pi$  de variance  $\propto C$

On simule une MCMC  $\theta_1, \dots, \theta_M \sim$  loi a posteriori  $\theta | y_i, \mathbf{x}_i$

On obtient  $M$  fonctions de prédiction  $(f_{\theta_1}, \dots, f_{\theta_M})$

Pour chaque  $\mathbf{x}_i$  on a  $M$  prédictions: une distribution de prédictions

On obtient une prédiction moyenne avec un intervalle de crédibilité

$C$  est simulé suivant un modèle hiérarchique avec  $C \sim$  hyperprior

1. Quantifie l'incertitude des prédictions: essentiel pour des applications sensibles (diagnostic médical, voitures autonomes ...)
2. Basé sur la simulation MCMC (lente en grande dimension / risque de divergence)

Best of both worlds:

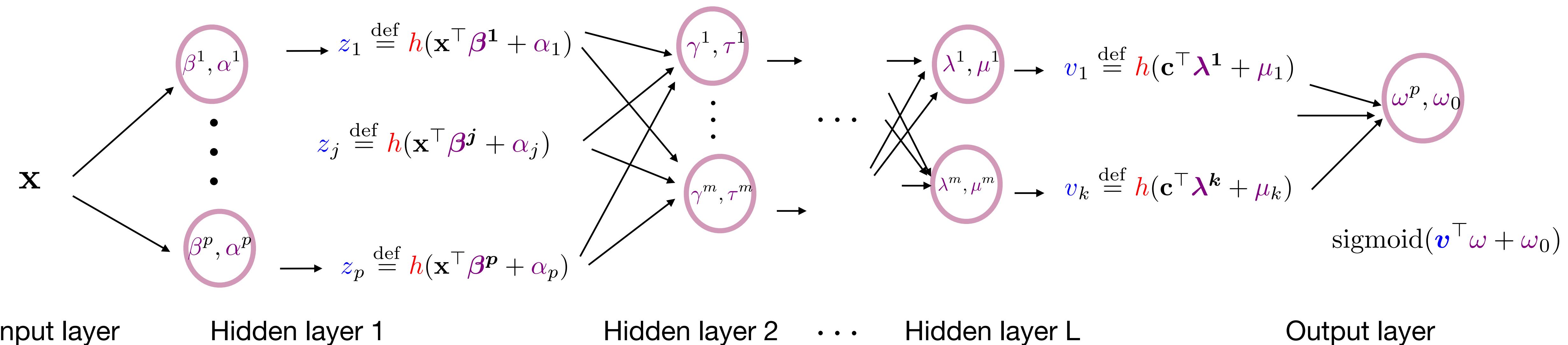
### 1. Classical ML with a bayesian flavor: *deep ensembles*

Entraîner  $m$  réseaux de neurones avec des initialisations différentes pour obtenir  $f_{\theta_1^*}, \dots, f_{\theta_m^*}$  et quantifier l'incertitude

### 2. Bayesian ML with a frequentist flavor: *bayesian output layer*

Optimiser tous les paramètres sauf les derniers:

Seules  $\omega$  et  $\omega_0$  sont aléatoires

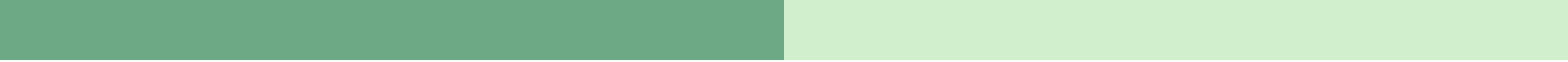


## 1. La modélisation bayésienne:

1. Utile pour incorporer des connaissances a priori
2. ...surtout lorsqu'on a peu de données
3. ...ou lorsqu'on a plusieurs groupes et une modélisation hiérarchique est pertinente
4. Converge vers l'approche fréquentiste lorsque la quantité de données est grande
5. Utile en ML pour avoir des prédictions avec des intervalles de crédibilité (incertitude)
6. Est faite en pratique avec des algorithmes de simulation (Monte-Carlo / Markov-Chain Monte-Carlo)
7. ... qui sont lents en grande dimension (nombre de paramètres)

## 2. L'approche fréquentiste:

1. Donne des estimations ponctuelles (ne quantifie pas l'incertitude)
2. Repose sur des algorithmes d'optimisation
3. ... qui sont rapides (relativement) et parallélisables
4. Utile pour un déploiement rapide à grande échelle (large data)



# What can you do next ?

## 1. Bayesian clustering (non-supervisé):

Identifier K groupes (clusters) dans les données en utilisant une variable latente (non-observée)

## 2. Non-parametric Bayesian clustering:

K inconnu: on suppose un a priori sur K avec un processus aléatoire (Dirichlet Process)

(Gelman 2013), Ch. 21, 22, 23

*Utilité: Segmentation des clients (profiling), données génomiques*



(Frazier, 2018)



## 3. Bayesian Optimization:

Optimiser une fonction complexe avec une exploration aléatoire guidée par un Processus Gaussien

*Utilité: Optimisation de l'erreur de validation croisée avec beaucoup de paramètres*

## 4. Variational methods:

Approximer la densité a posteriori avec la densité paramétrique la plus proche (typiquement gaussienne)

(Gelman 2013), Ch 13.7



*Utilité: MCMC sont lents en grande dimension: retour à l'optimisation*

## 5. Sequential Monte-Carlo (SMC) / Particle filters / Hidden Markov Models:

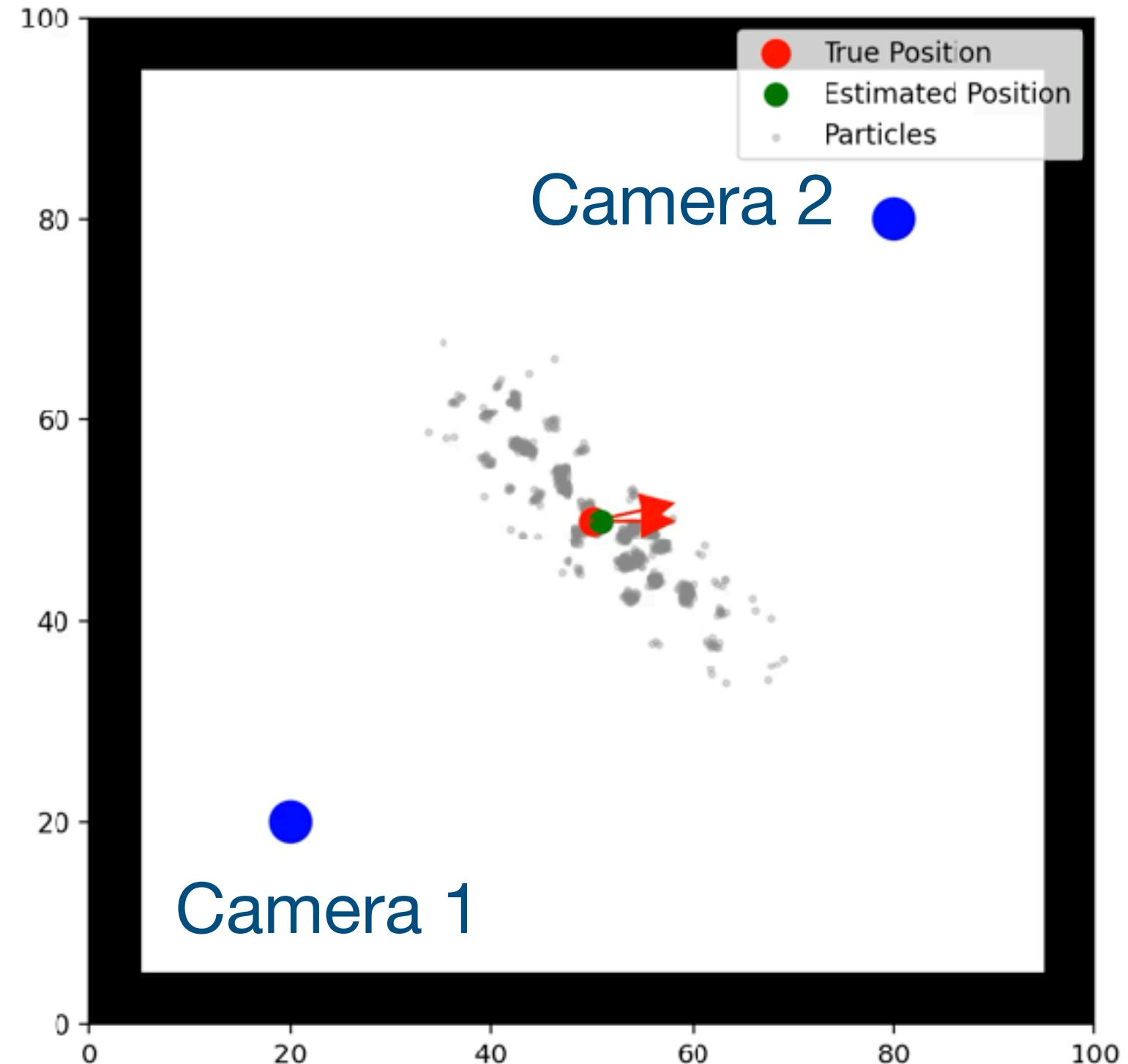
Approximer des distributions “in real-time” avec des particules pondérées qui sont mises à jour

(Doucet 2001), intro

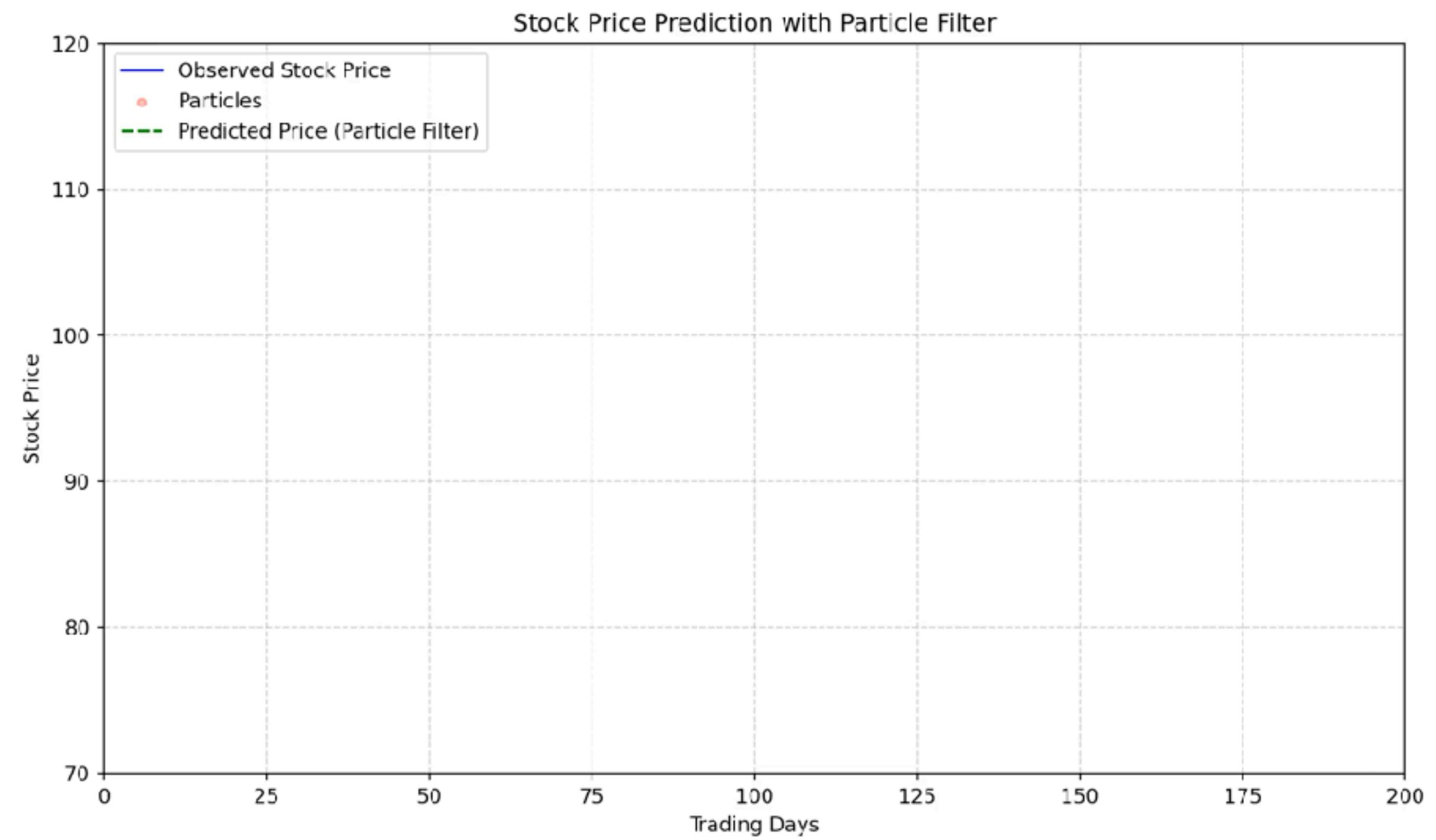


*Utilité: Real-time Tracking, Time series forecasting*

## Real time tracking:



## Stock prices prediction:



## 5. Sequential Monte-Carlo (SMC) / Particle filters / Hidden Markov Models:

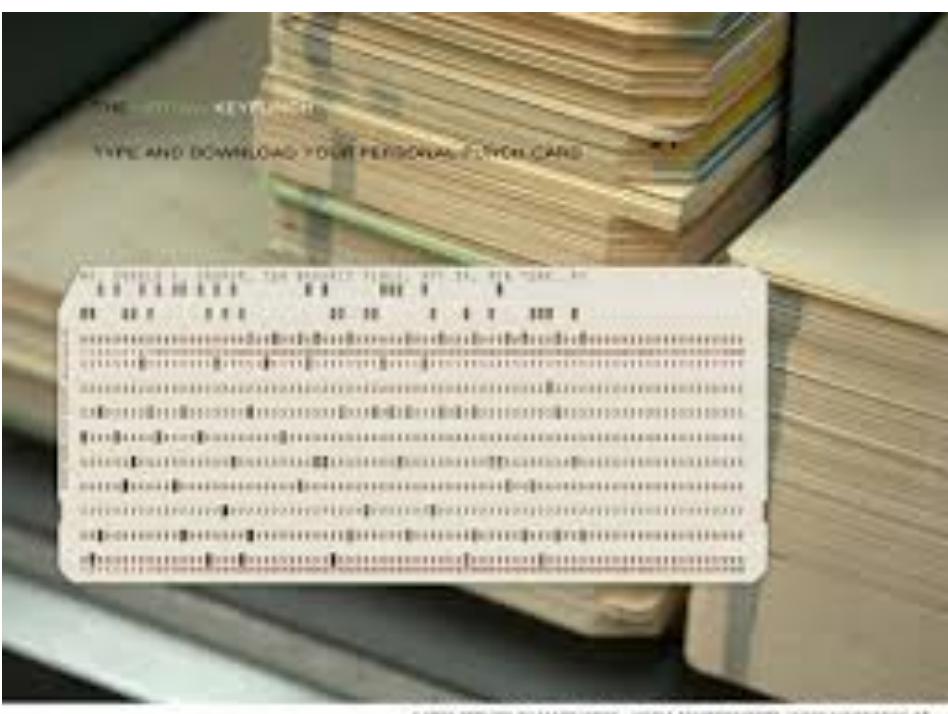
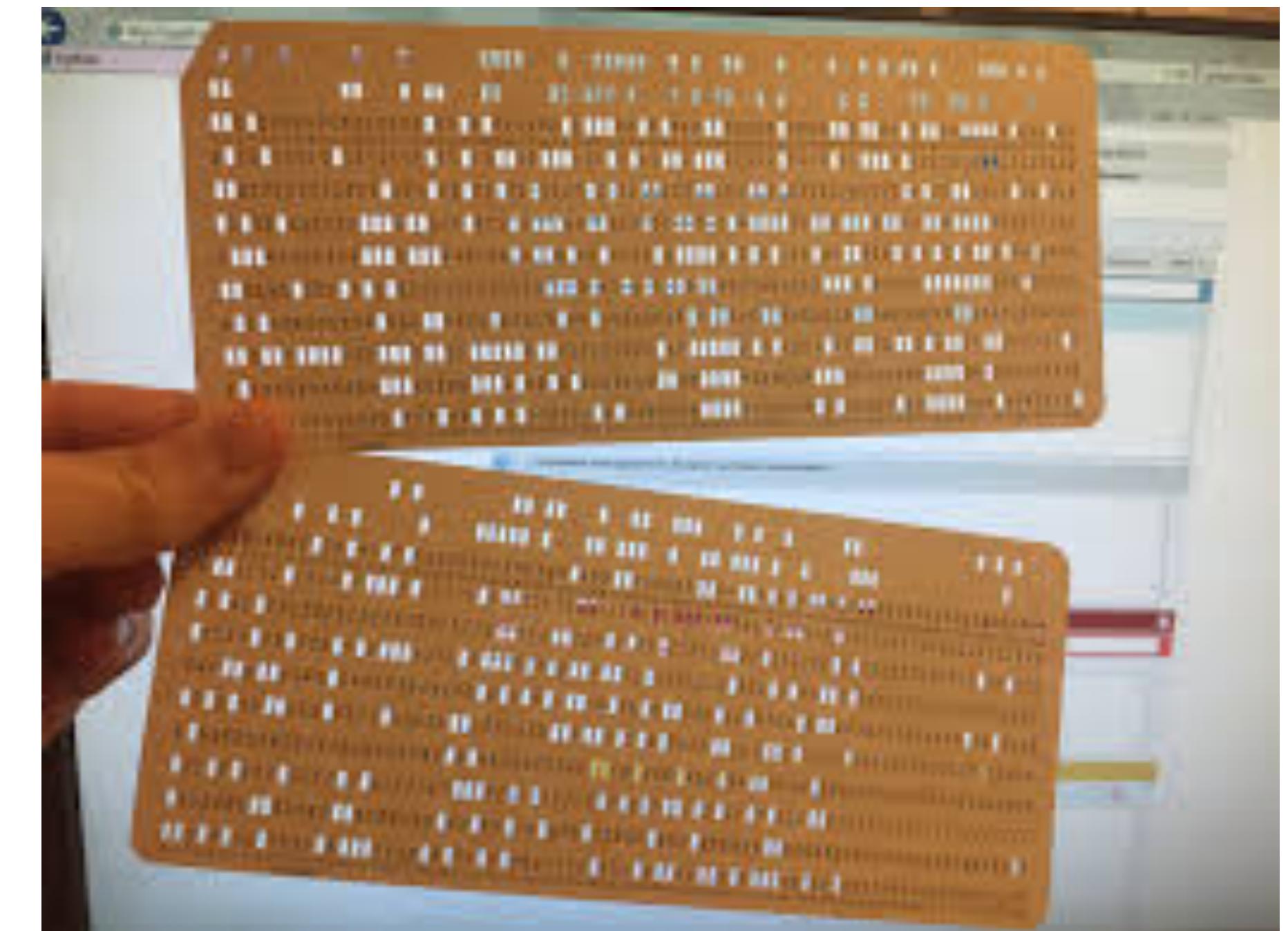
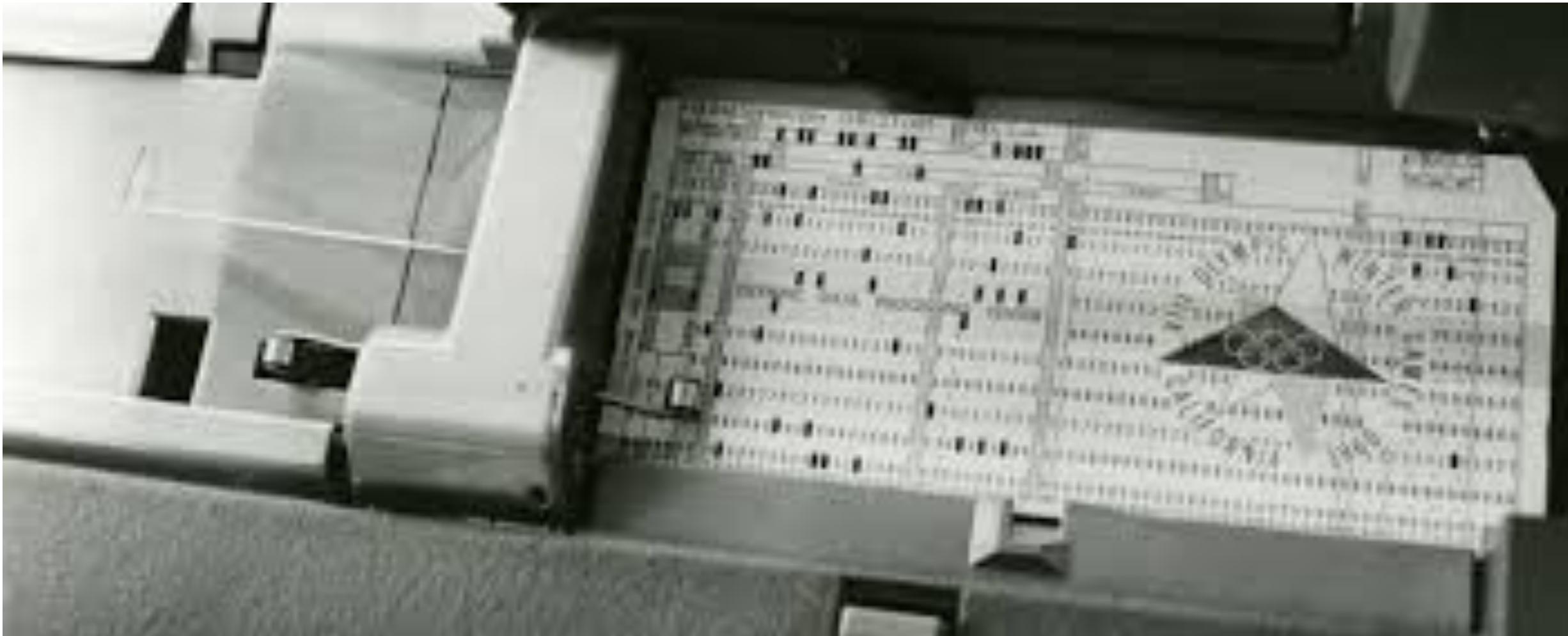
Approximer des distributions “in real-time” avec des particules pondérées qui sont mises à jour

*Utilité: Real-time Tracking, Time series forecasting*

# AI & Coding

# Programming in the 70s

## IBM Punch cards (COBOL)



When compilers came, experts thought it was the end of “programers” .. the opposite happened

Will it be the same with AI ?

Nobody knows

# AMA: Ask me anything