

Chapitre IV: Introduction to representation learning

Hicham Janati

hjanati@insea.ac.ma



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 prédition.

Si elle prédit que le client A va résilier son abonnement dans un mois, elle peut:

1. Le contacter pour essayer de comprendre sa situation et lui faire changer d'avis
2. Le cibler avec des offres de promotion

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	?

$\mathbf{X}^1 \quad \mathbf{X}^2 \quad \mathbf{X}^3 \quad \mathbf{X}^4 \quad \mathbf{X}^5 \quad \mathbf{X}^6 \quad \mathcal{Y}$

On modélise cette base de données par des variables aléatoires

On veut utiliser le vecteur aléatoire $\mathbf{X} = (\mathbf{X}^1, \dots, \mathbf{X}^6)^\top$ pour prédire le \mathcal{Y}

On cherche une fonction $f : \mathbb{R}^6 \rightarrow \{0, 1\}$ telle que: $f(\mathbf{X}) = \mathcal{Y}$

Ici: $\mathcal{Y} \in \{0, 1\}$ v.a discrète: Problème de classification

En revanche, si on avait: $\mathcal{Y} \in \mathbb{R}$ v.a continue: Problème de régression

Soit \mathbf{X} un vecteur aléatoire dans \mathbb{R}^d

Soit y une variable aléatoire dans \mathcal{Y}

L'ensemble \mathcal{Y} peut être fini (ex. $\{0, 1\}$ – classification) ou infini (ex. \mathbb{R} – régression)

On cherche une fonction $f : \mathbb{R}^d \rightarrow \mathcal{Y}$ telle que: $f(\mathbf{X}) = y$

On note l'écart entre la prédiction $f(\mathbf{X})$ et les vrais labels y par $\mathcal{L}(f(\mathbf{X}), y)$

\mathcal{L} est appelée: fonction de perte – *loss function*

Exemples:

$$\mathcal{Y} = \mathbb{R}$$

$$\mathcal{L}(f(\mathbf{X}), y) = (f(\mathbf{X}) - y)^2$$

Squared loss

$$\mathcal{L}(f(\mathbf{X}), y) = |f(\mathbf{X}) - y|$$

Absolute loss

$$\mathcal{Y} = \{0, 1\}$$

$$\mathcal{L}(f(\mathbf{X}), y) = \mathbb{1}(f(\mathbf{X}) \neq y)$$

0-1 loss

Soit \mathcal{F} l'ensemble des fonctions $\mathbb{R}^d \rightarrow \mathcal{Y}$

On veut une fonction $f \in \mathcal{F}$ qui minimise la perte $\mathcal{L}(f(\mathbf{X}), y)$

Or \mathbf{X} et y sont des variables aléatoires, on minimise alors:

$$\min_{f \in \mathcal{F}} \mathbb{E}(\mathcal{L}(f(\mathbf{X}), y))$$

Expected Risk Minimization

$$\min_{f \in \mathcal{F}} \mathbb{E}(\mathcal{L}(f(\mathbf{X}), y))$$

Expected Risk Minimization

Que doit-on faire pour résoudre ce problème ?

1. Prendre f dans un sous-ensemble $\mathcal{H} \subset \mathcal{F}$ de dimension finie
2. Avoir des observations $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \sim \mathbb{P}(\mathbf{X}, Y)$
3. Minimiser le risque empirique $\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^n (\mathcal{L}(f(\mathbf{x}_i), y_i))$
avec un algorithme d'optimisation

1. Modélisation

Comment choisir \mathcal{H} ?

Comment choisir \mathcal{L} ?

2. Approximation statistique

Les observations sont-elles i.i.d. ?

n est-il assez grand ?

3. Optimisation numérique

La solution existe ? est unique ?

Quel algorithme d'optimisation ?

1. Modélisation

Comment choisir \mathcal{H} ?

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

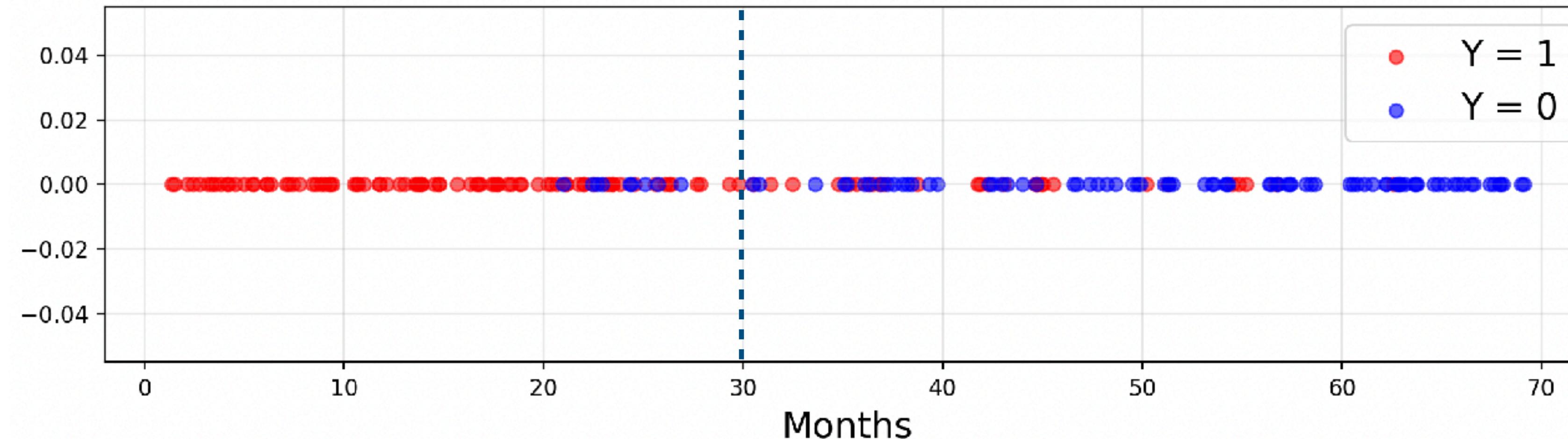
L'entreprise souhaite anticiper le “churn” avec un algorithme de prédiction.

La fonction de prédiction f doit donner 1 ou 0, on considère alors des fonctions de type:

$$f(\mathbf{x}) = \mathbf{1}_{g(\mathbf{x}) \geq 0}$$

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 “ $x = \text{Months}$ ” qui donne la durée du contrat:



Fonction de prédiction: $f(\mathbf{x}) = \mathbf{1}_{g(\mathbf{x}) \geq 0}$

Proposer une fonction g simple telle que f distingue au mieux les labels en moyenne

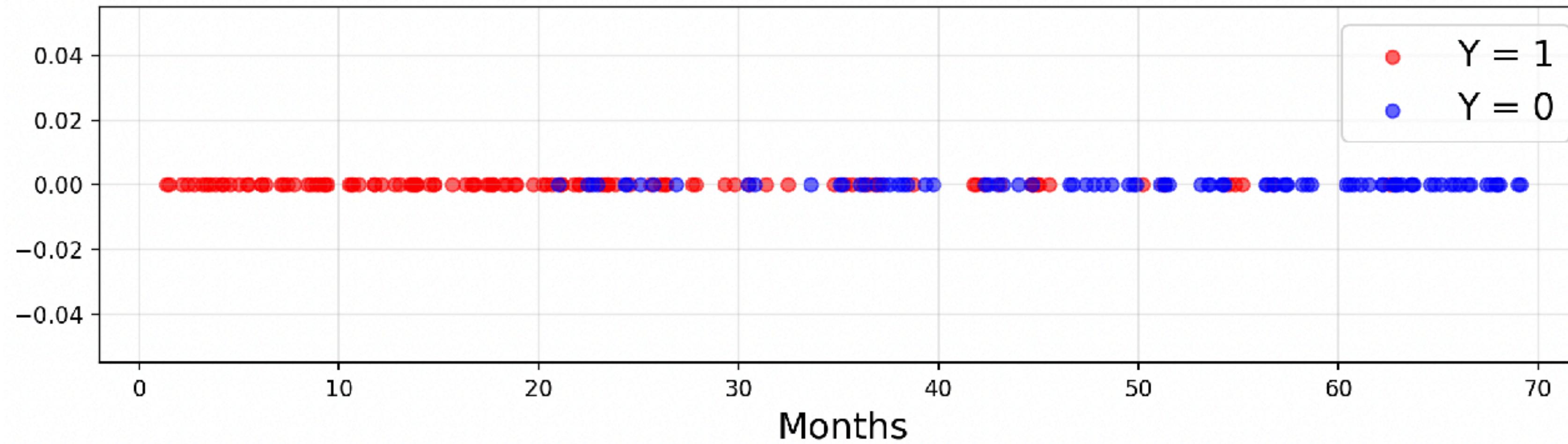
Par ex: $g(\mathbf{x}) = -\mathbf{x} + 30$

$$f(\mathbf{x}) = \mathbf{1}(g(\mathbf{x}) \geq 0) = \mathbf{1}(-\mathbf{x} + 30 \geq 0) = \mathbf{1}(\mathbf{x} \leq 30)$$

On peut par exemple considérer la famille des fonctions linéaires: $g(\mathbf{x}) = \beta_1 \mathbf{x} + \beta_0$

On dit que g est paramétrée par $\beta = (\beta_0, \beta_1)^\top$

On considère une seule variable “ $x = \text{Months}$ ” qui donne la durée du contrat:



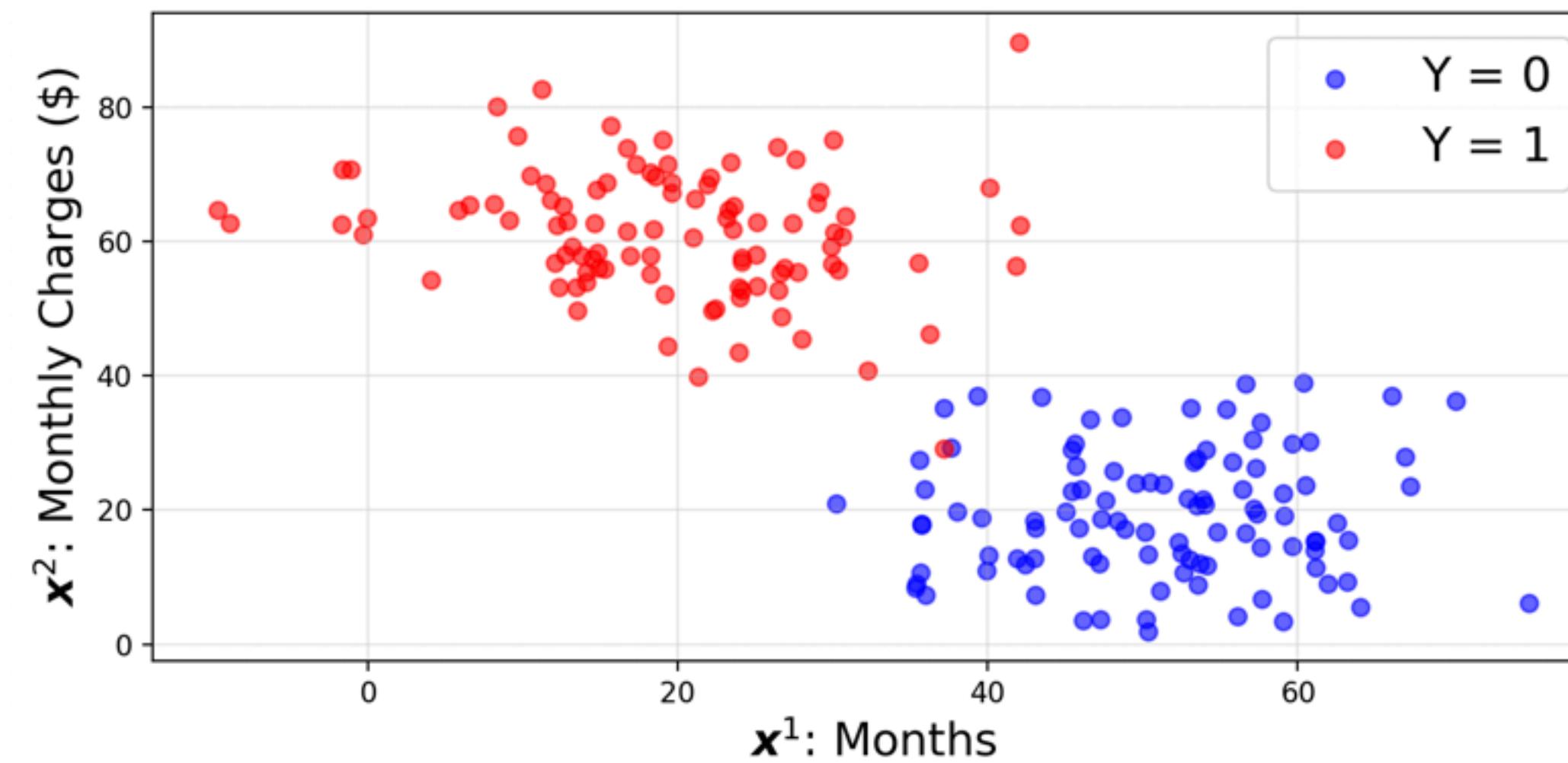
Ainsi, $\mathcal{H} = \{f : \mathbf{x} \mapsto \mathbb{1}(\beta_1 \mathbf{x} + \beta_0 \geq 0), \beta_0, \beta_1 \in \mathbb{R}\}$

Chercher la meilleure f = chercher le meilleur β :

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

On considère deux variables: "Months" et "MonthlyCharges":

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



Quelle serait la fonction paramétrée g la plus simple ici ?

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

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

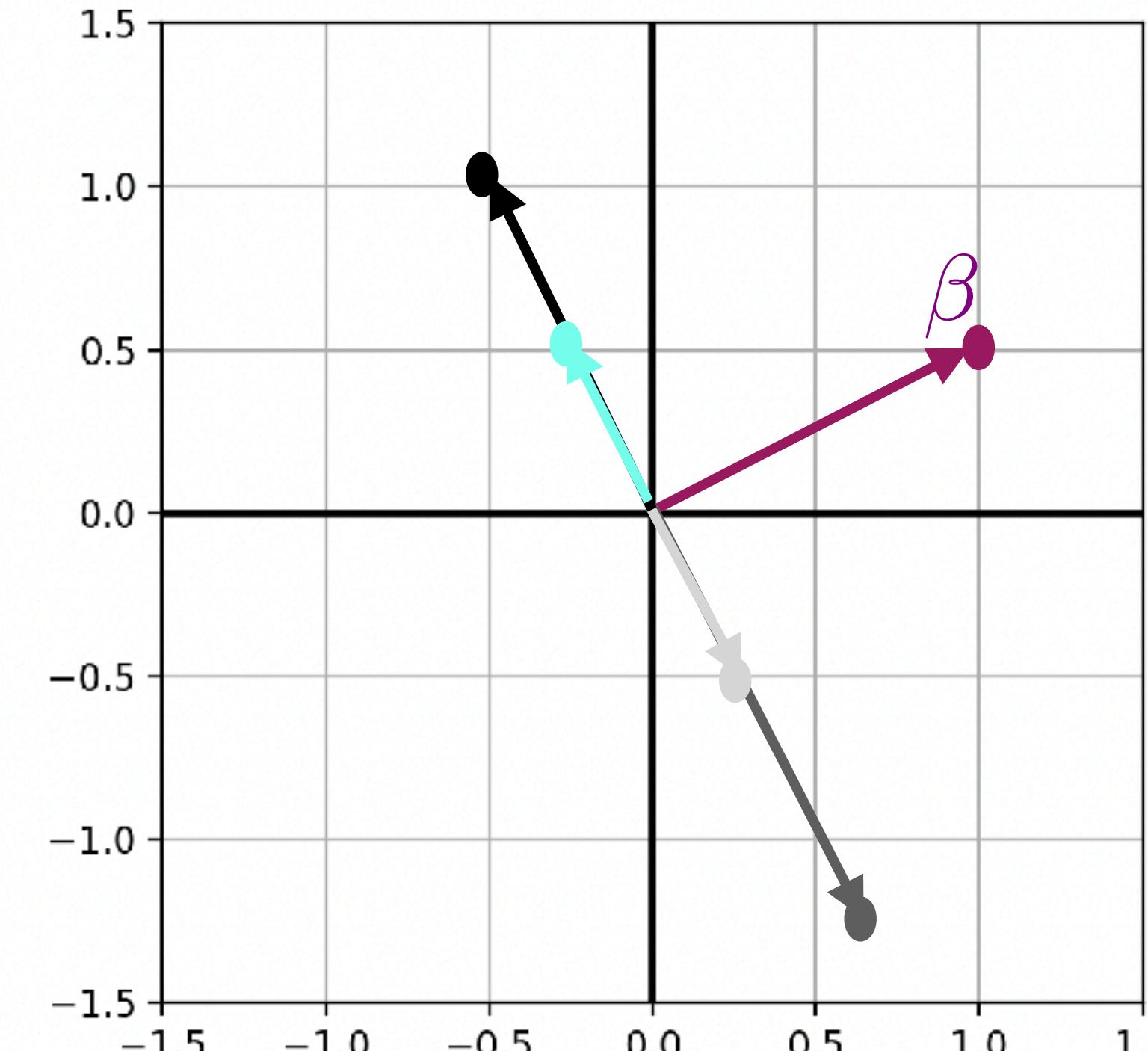
$$g(\mathbf{x}) = \alpha + \beta^\top \mathbf{x}, \quad \alpha \in \mathbb{R}, \beta \in \mathbb{R}^2$$

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

À quoi ressemble l'ensemble des fonctions g graphiquement ?

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



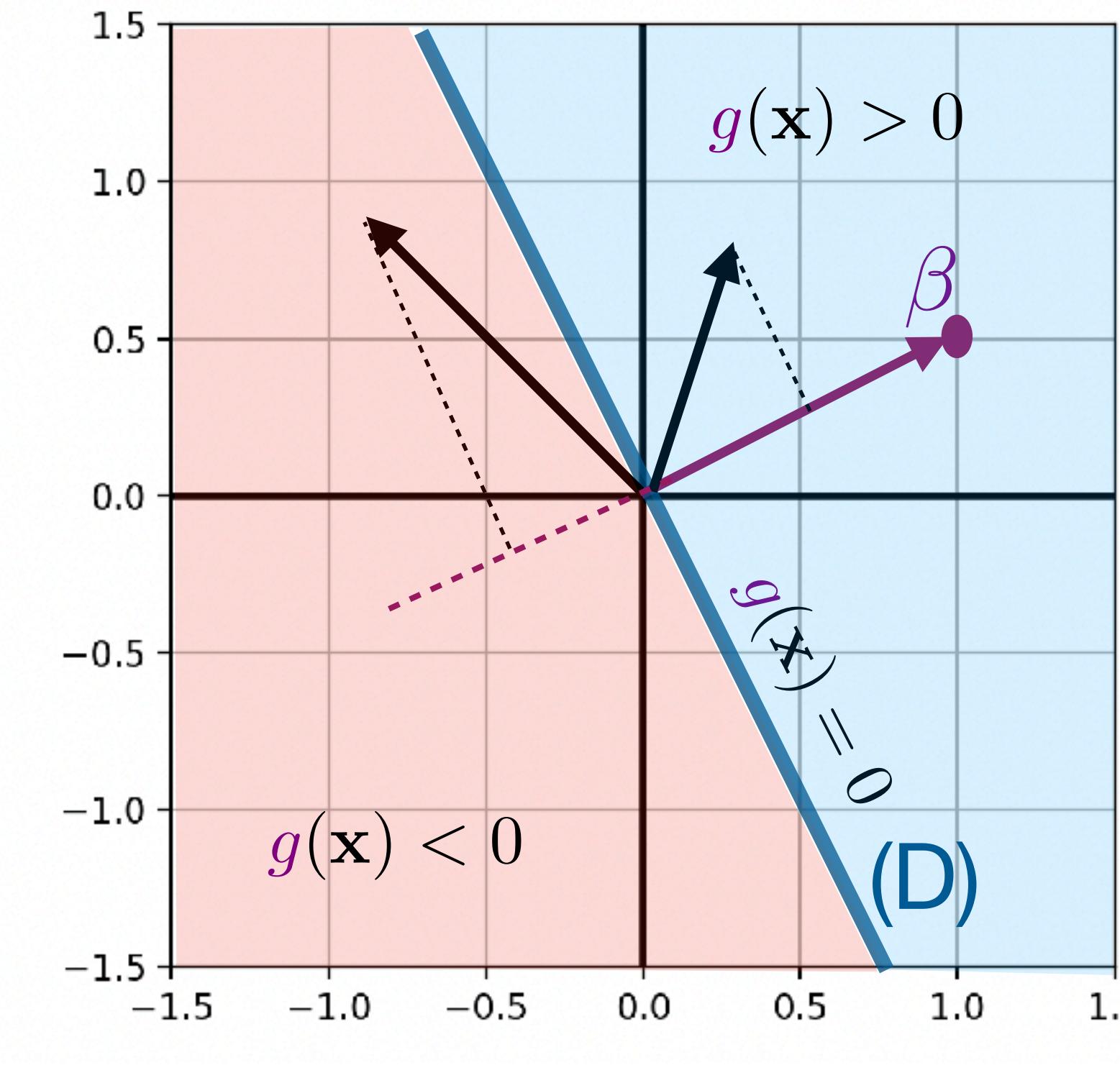
Exemple avec $\beta = (1, 0.5)^\top$ et $c = 0$.

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

Tous les vecteurs orthogonaux à β .

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

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 $\beta^\top \mathbf{x} = 0$?

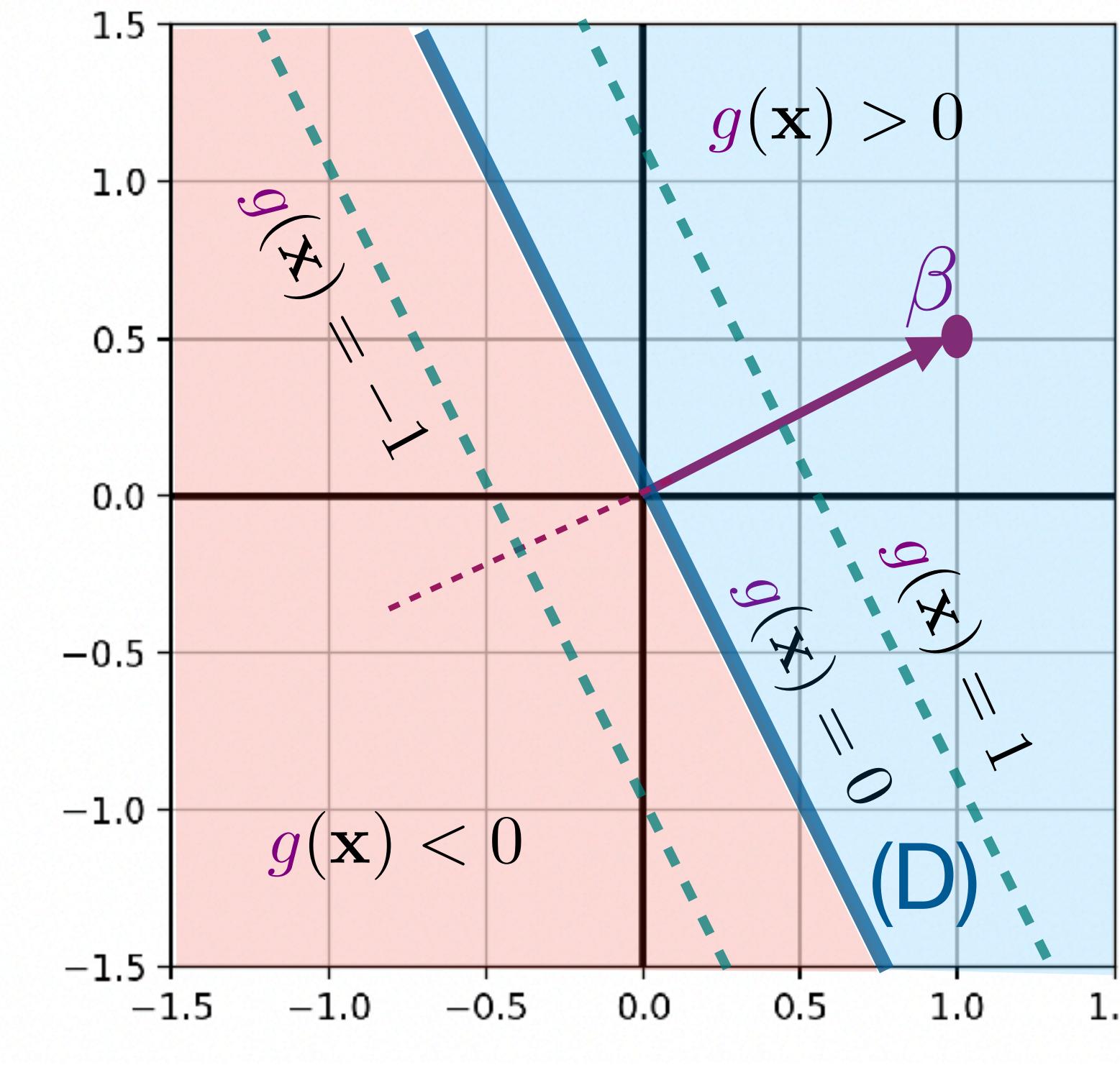
Tous les vecteurs orthogonaux à β .

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

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 $c = 0$.

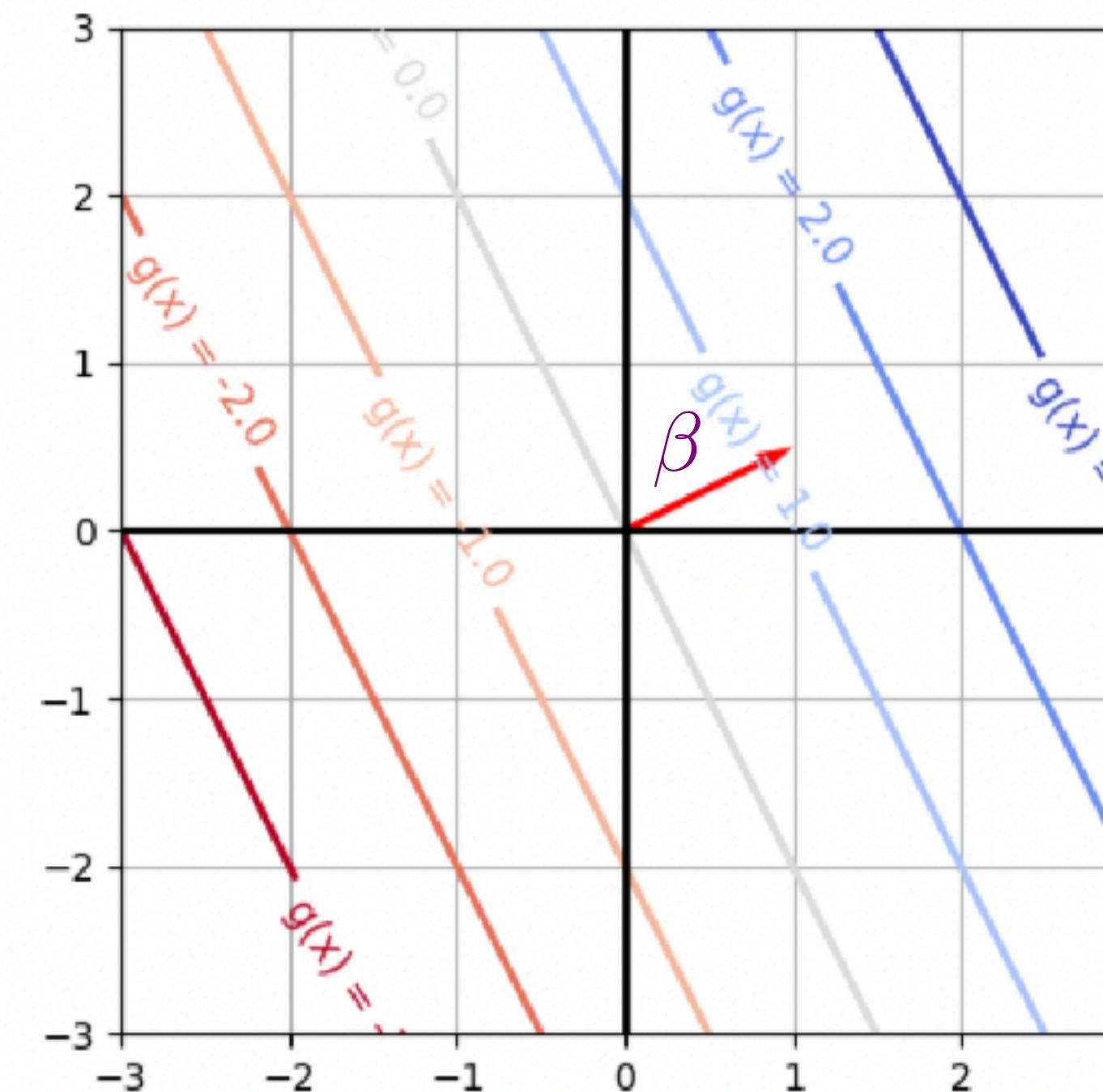
Quels sont les \mathbf{x} tels que $\beta^\top \mathbf{x} = 0$?

Tous les vecteurs orthogonaux à β .

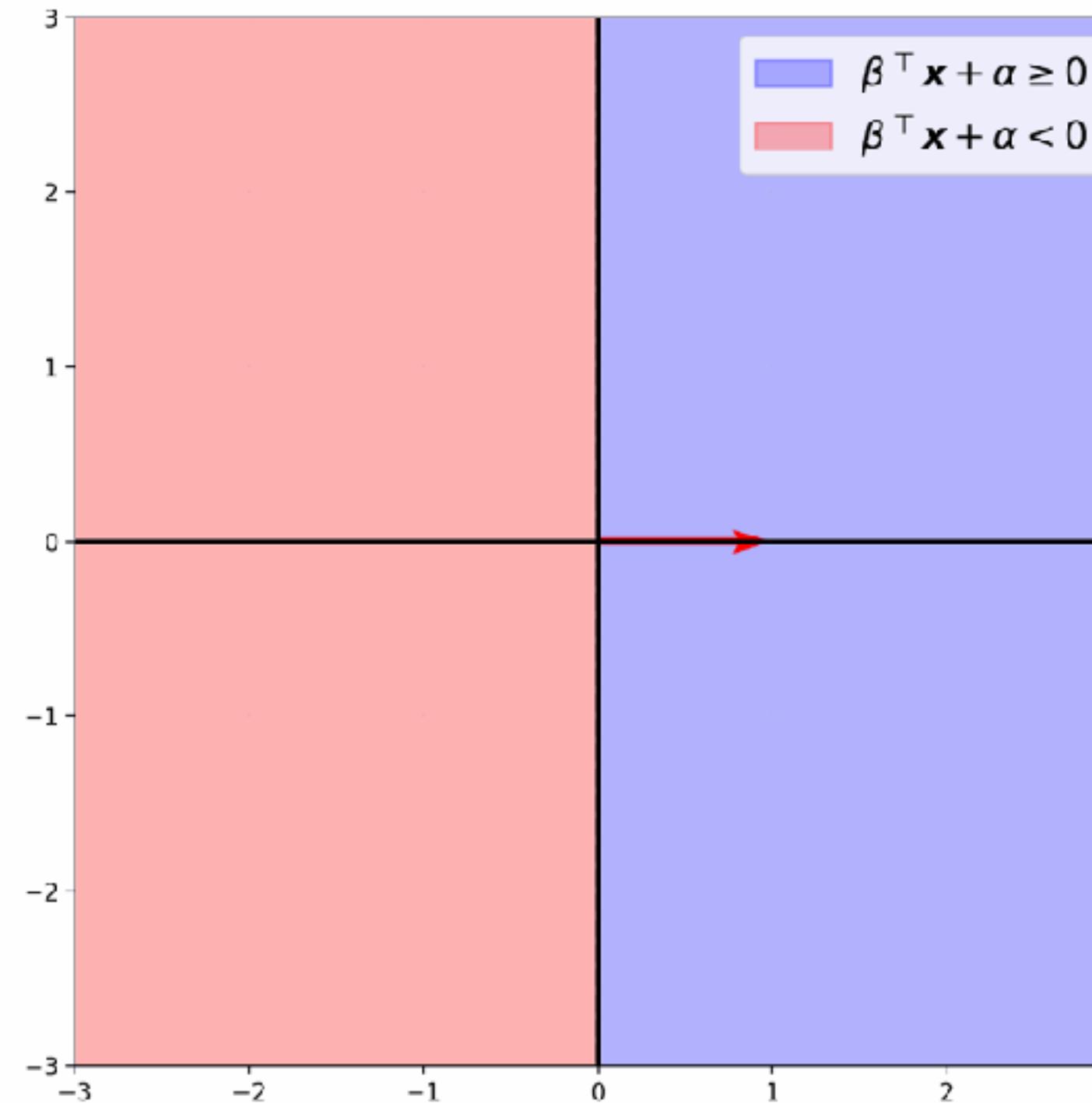
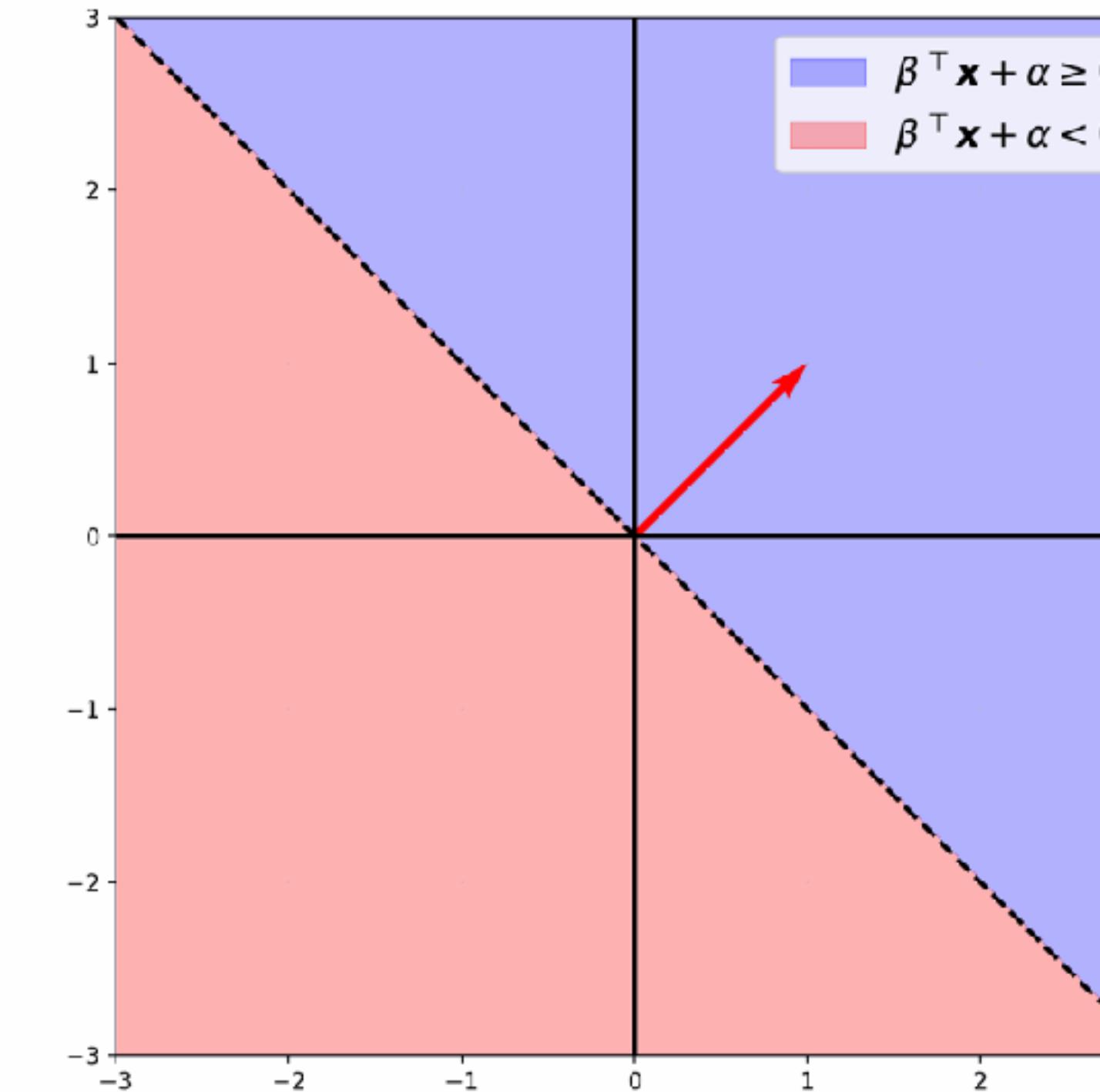
$\{\mathbf{x} \in \mathbb{R}^2 | \beta^\top \mathbf{x} = 0\}$ est la droite perpendiculaire à β .

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

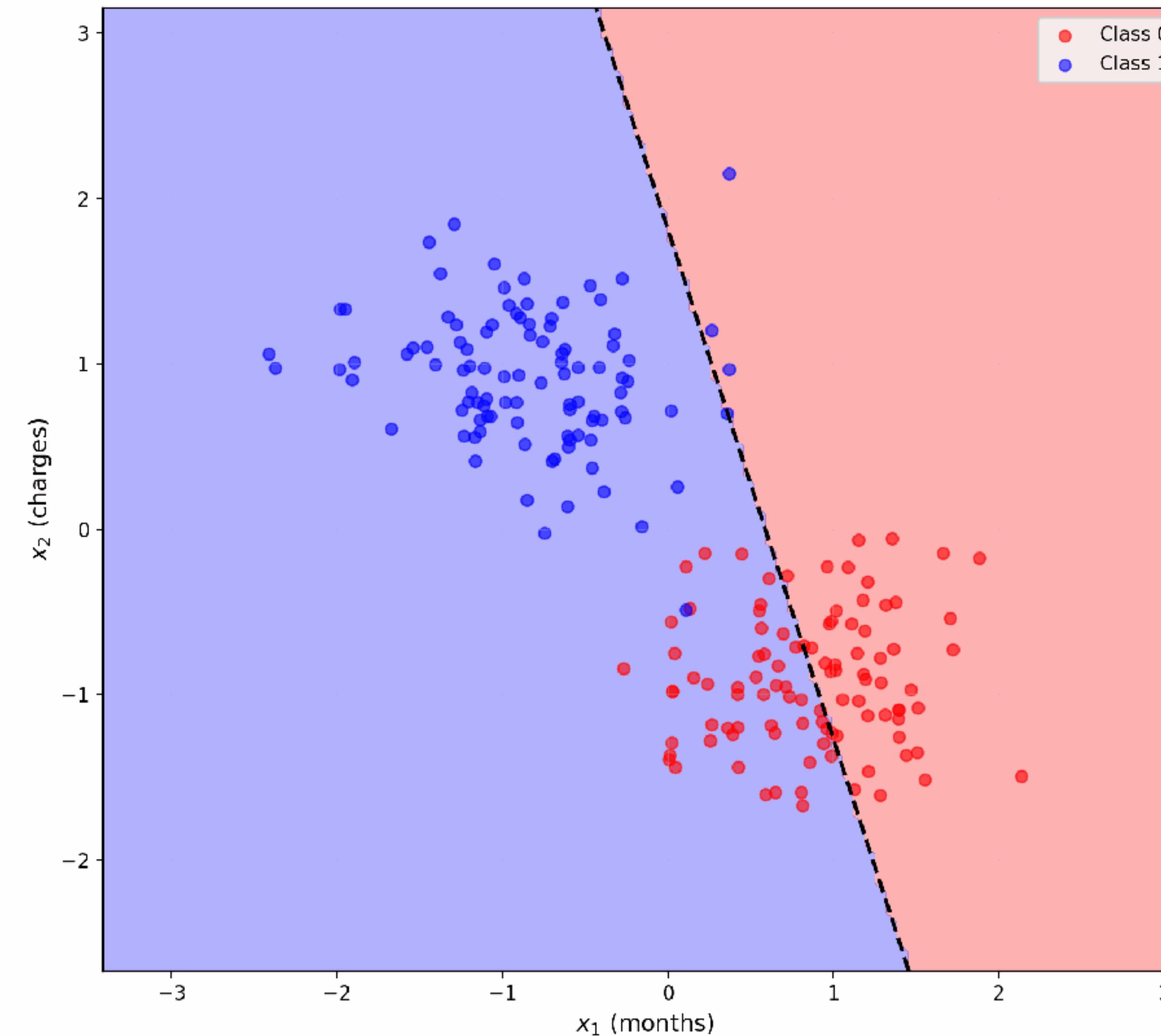
à gauche de (D), $\beta^\top \mathbf{x} < 0$



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

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

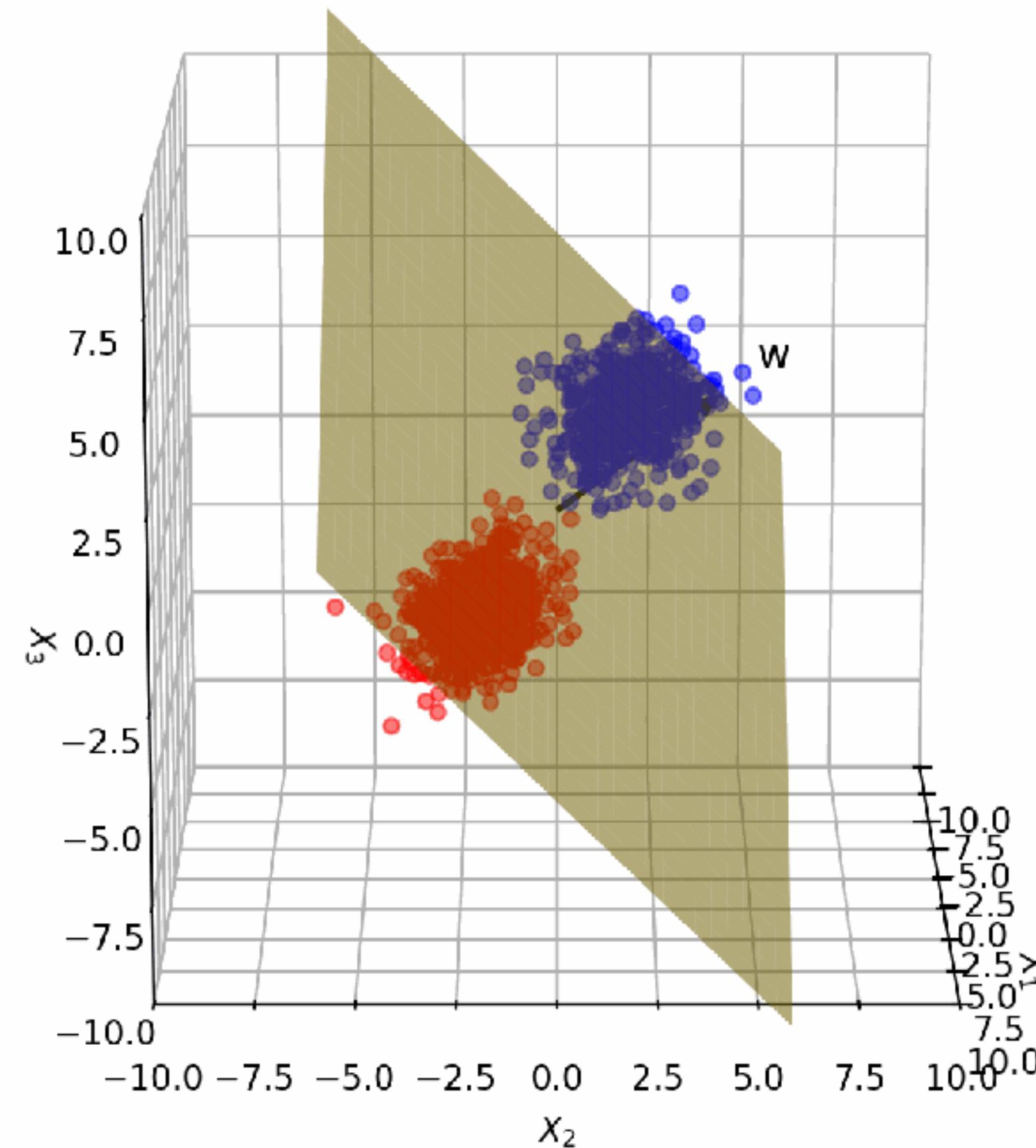
$$\min_{\alpha \in \mathbb{R}, \beta \in \mathbb{R}^2} \sum_{i=1}^n \mathcal{L}(\mathbf{1}_{\{\alpha + \beta^\top \mathbf{x}_i \geq 0\}}, y_i)$$



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

Supposons on a deux observations $\mathbf{x}_1, \mathbf{x}_2$ avec le vrai label $y_1 = y_2 = 1$ et:

$$\alpha + \beta^\top \mathbf{x}_1 = 140.23 \quad \alpha + \beta^\top \mathbf{x}_2 = 0.1$$

Quelle est la loss associée $\mathcal{L}(f(\mathbf{x}_i), y_i)$ à chacune de ces prédictions ?

Les prédictions sont données par $f(\mathbf{x}_i) = \mathbb{1}(\alpha + \beta^\top \mathbf{x}_i \geq 0) = 1$

La prédiction est correcte: la **loss** est 0 dans les deux cas !

Or on aimeraient un modèle où la prédiction de \mathbf{x}_1 est plus confiante que \mathbf{x}_2

Idée: transformer le score $\alpha + \beta^\top \mathbf{x}_i$ en une probabilité

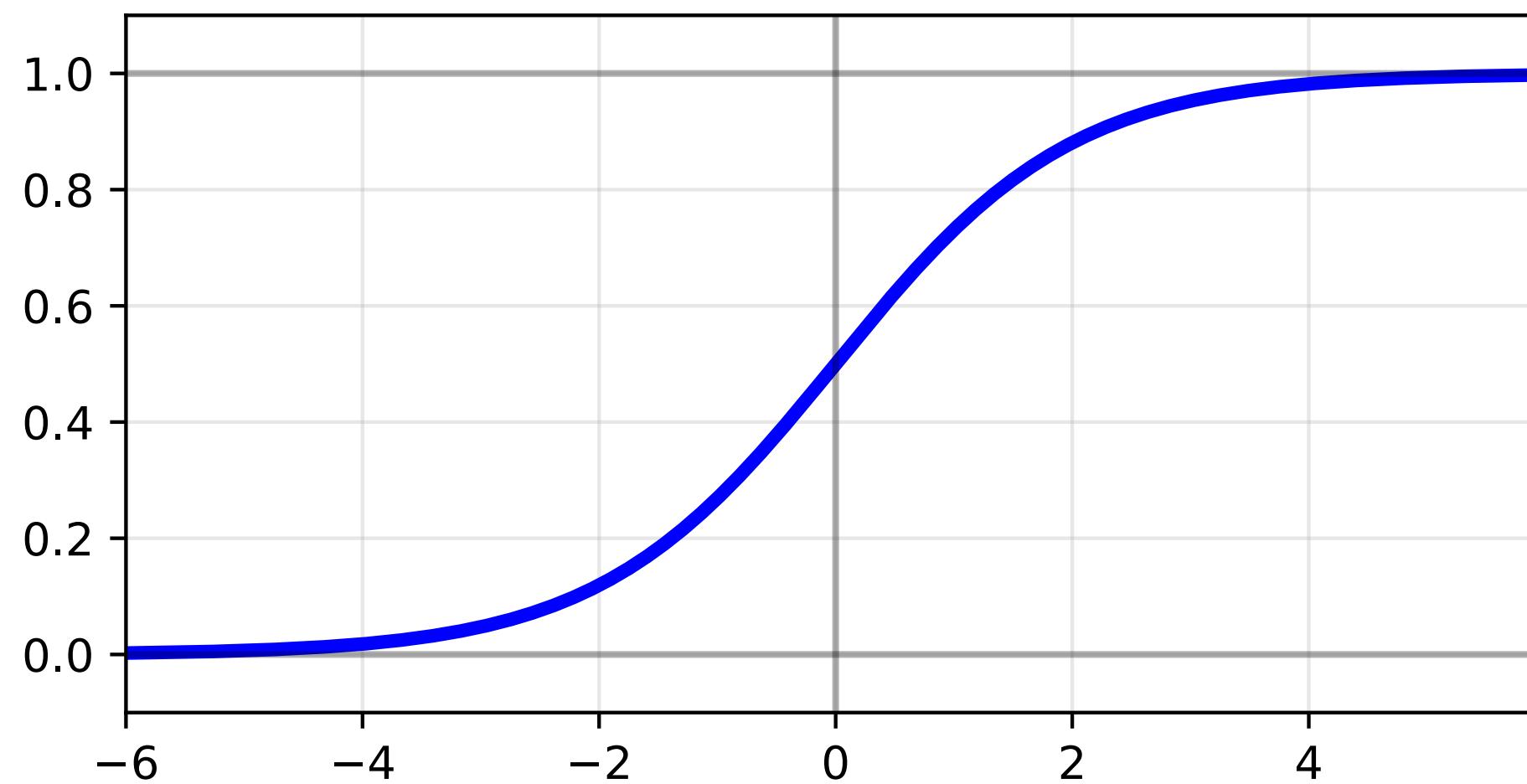
$$\alpha + \beta^\top \mathbf{x}_i = 1044.2 \Rightarrow \mathbb{P}(y_i = 1 | \mathbf{x}_i) = 0.999$$

$$\alpha + \beta^\top \mathbf{x}_i = 0.24 \Rightarrow \mathbb{P}(y_i = 1 | \mathbf{x}_i) = 0.51$$

$$\alpha + \beta^\top \mathbf{x}_i = -946.5 \Rightarrow \mathbb{P}(y_i = 1 | \mathbf{x}_i) = 0.001$$



On peut utiliser la fonction sigmoid: $\sigma : t \mapsto \frac{\exp(t)}{1+\exp(t)}$



$$\sigma(+\infty) = 1$$

$$\sigma(-\infty) = 0$$

$$\sigma(0) = \frac{1}{2}$$

Et on modélise les probabilités: $\mathbb{P}(y_i = 1 | \mathbf{x}_i) = \sigma(\alpha + \beta^\top \mathbf{x}_i)$

Au lieu d'avoir des prédictions binaires uniquement, nous avons des probabilités $p_i = \sigma(\alpha + \beta^\top \mathbf{x}_i)$

On veut comparer les p_i et les y_i avec une **loss** \mathcal{L} . Comment doit-elle se comporter selon les cas suivants:

$$y_i = 1 \quad \text{et}$$

- (a) $p_i \rightarrow 1 \Rightarrow$ Perte $\mathcal{L}(p_i, y_i) \rightarrow 0$
- (b) $p_i \rightarrow 0 \Rightarrow$ Perte $\mathcal{L}(p_i, y_i) \rightarrow +\infty$

Quelle fonction $\mathcal{L}(p_i, 1)$ vérifie cela ?

$$-\log(p_i)$$

$$y_i = 0 \quad \text{et}$$

- (a) $p_i \rightarrow 1 \Rightarrow$ Perte $\mathcal{L}(p_i, y_i) \rightarrow +\infty$
- (b) $p_i \rightarrow 0 \Rightarrow$ Perte $\mathcal{L}(p_i, y_i) \rightarrow 0$

$$-\log(1 - p_i)$$

Quelle fonction $\mathcal{L}(p_i, 0)$ vérifie cela ?

Comment peut-on unifier les deux et définir $\mathcal{L}(p_i, y_i)$?

$$\mathcal{L}(p_i, y_i) = -y_i \log(p_i) - (1 - y_i) \log(1 - p_i)$$

Cross-entropy loss



$$\sigma : t \mapsto \frac{\exp(t)}{1 + \exp(t)}$$

$$p_i = \sigma(\alpha + \beta^\top \mathbf{x}_i) \quad \mathcal{L}(p_i, y_i) = -y_i \log(p_i) - (1 - y_i) \log(1 - p_i)$$

sigmoid / logistic
function

$$\min_{\alpha \in \mathbb{R}, \beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(p_i, y_i)$$



Optimisation numérique

$$\alpha^*, \beta^*$$

Cette optimisation est l'étape d'apprentissage ou d'entraînement
“learning” / “training” / “data fitting”

Modèle de régression logistique

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

“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

→ Comment évaluer ces prédictions ?

$$\frac{1}{n} \sum_{i=1}^n \mathbb{1}(f^*(\mathbf{x}_i) \neq y_i)$$

“Train” error

La train error est **optimisée**: on a littéralement cherché la meilleure fonction telle que

$$f(\mathbf{x}_i) = y_i$$

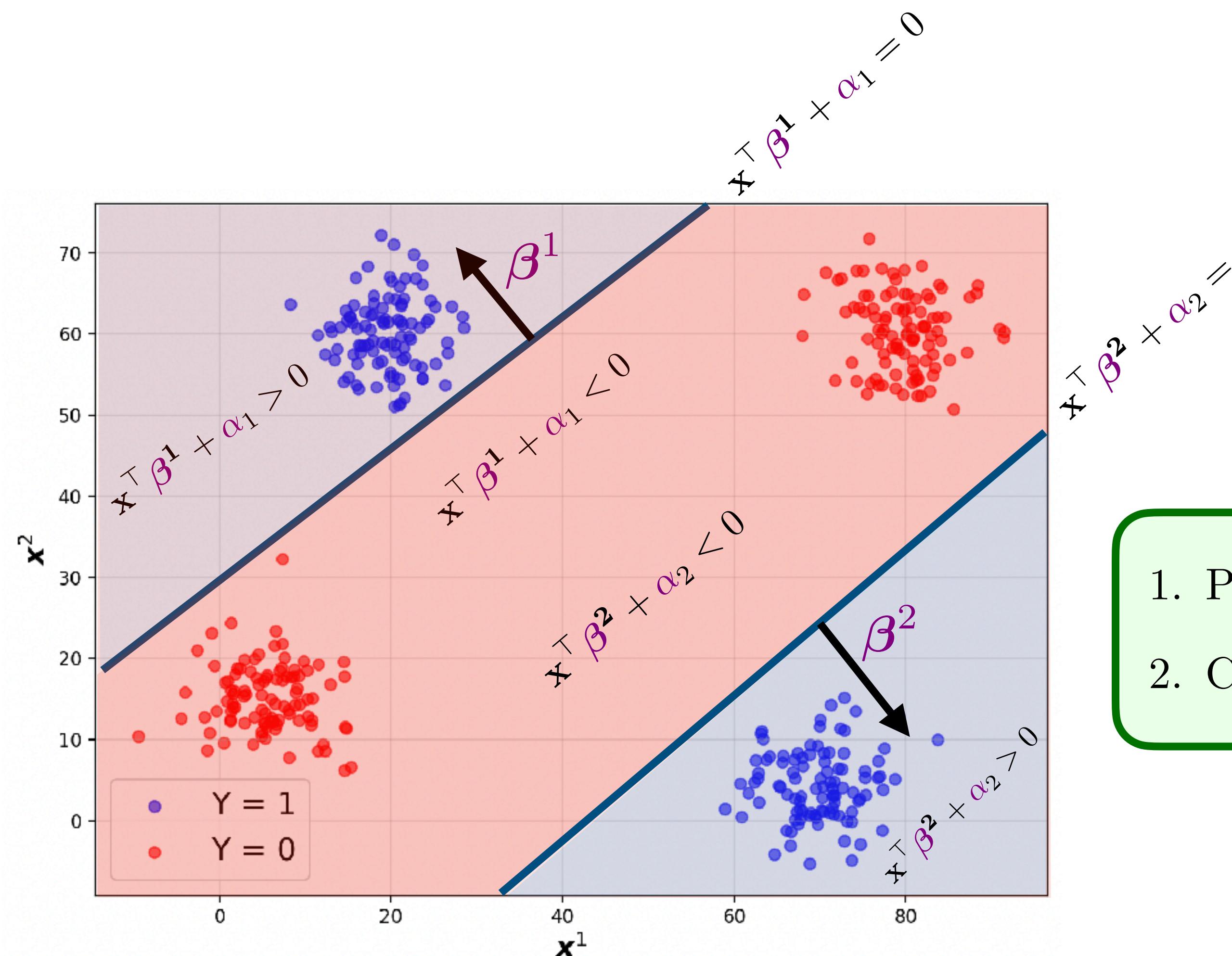
Il faut évaluer la performance du modèle sur des données **nouvelles** non vues à l’entraînement: “Test data”

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

→ “Test” error



Et si les données ressemblent à ceci ?



Aucune fonction linéaire ne peut séparer les classes

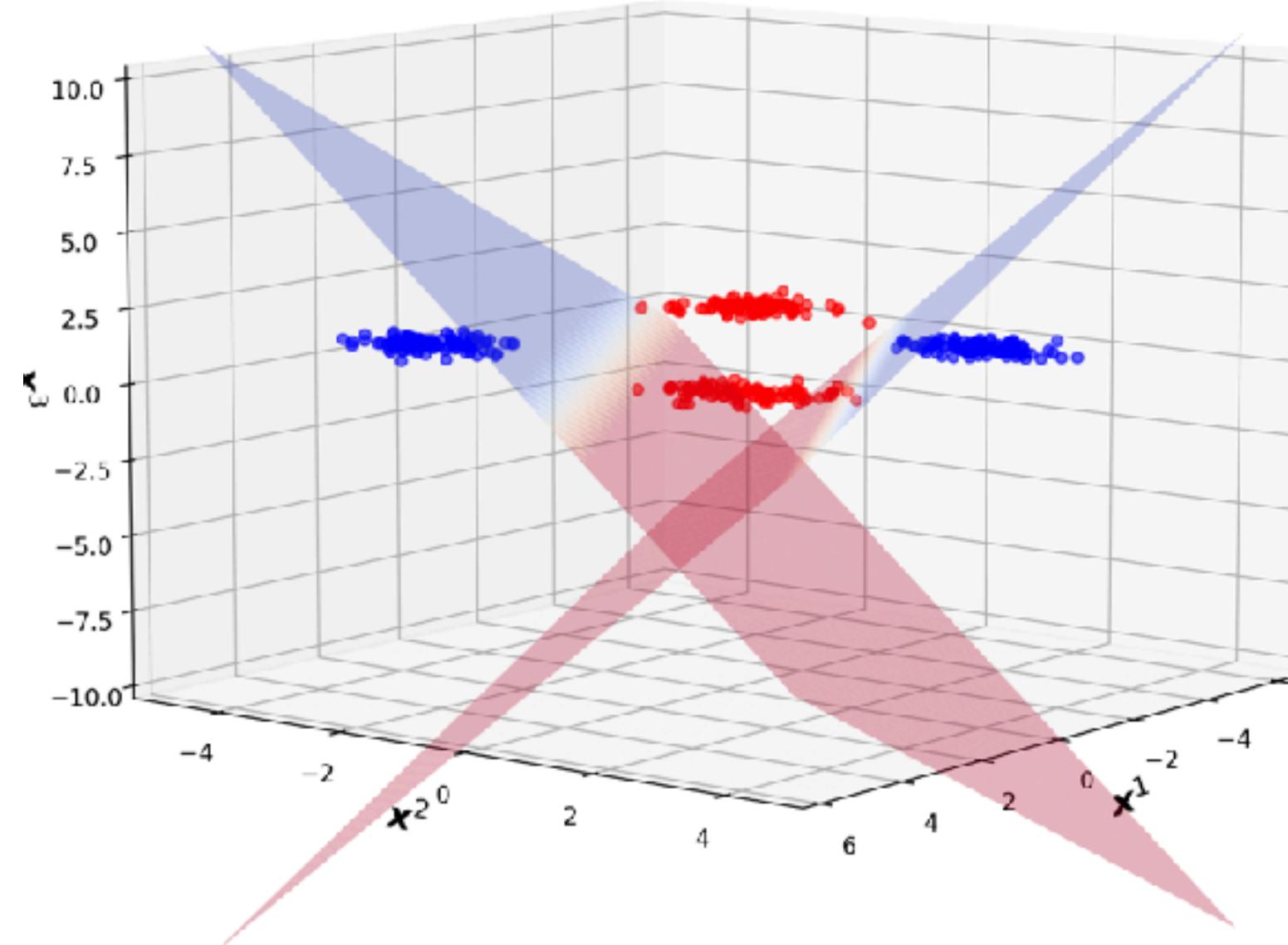
Idée: “combiner” plusieurs fonctions linéaires

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

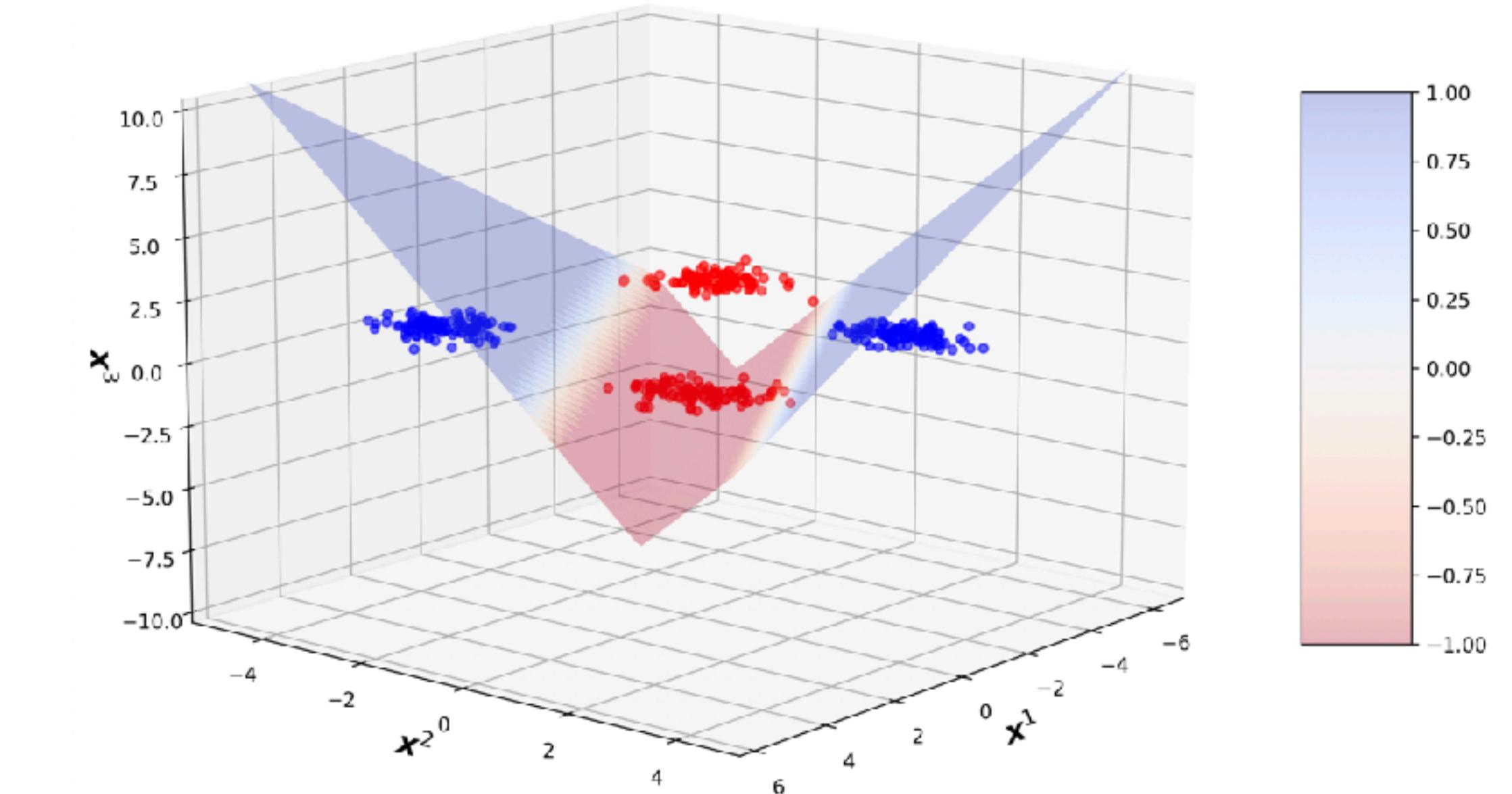
$$z_2 \stackrel{\text{def}}{=} \mathbf{x}^\top \boldsymbol{\beta}^2 + \alpha_2$$

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 α, β ?

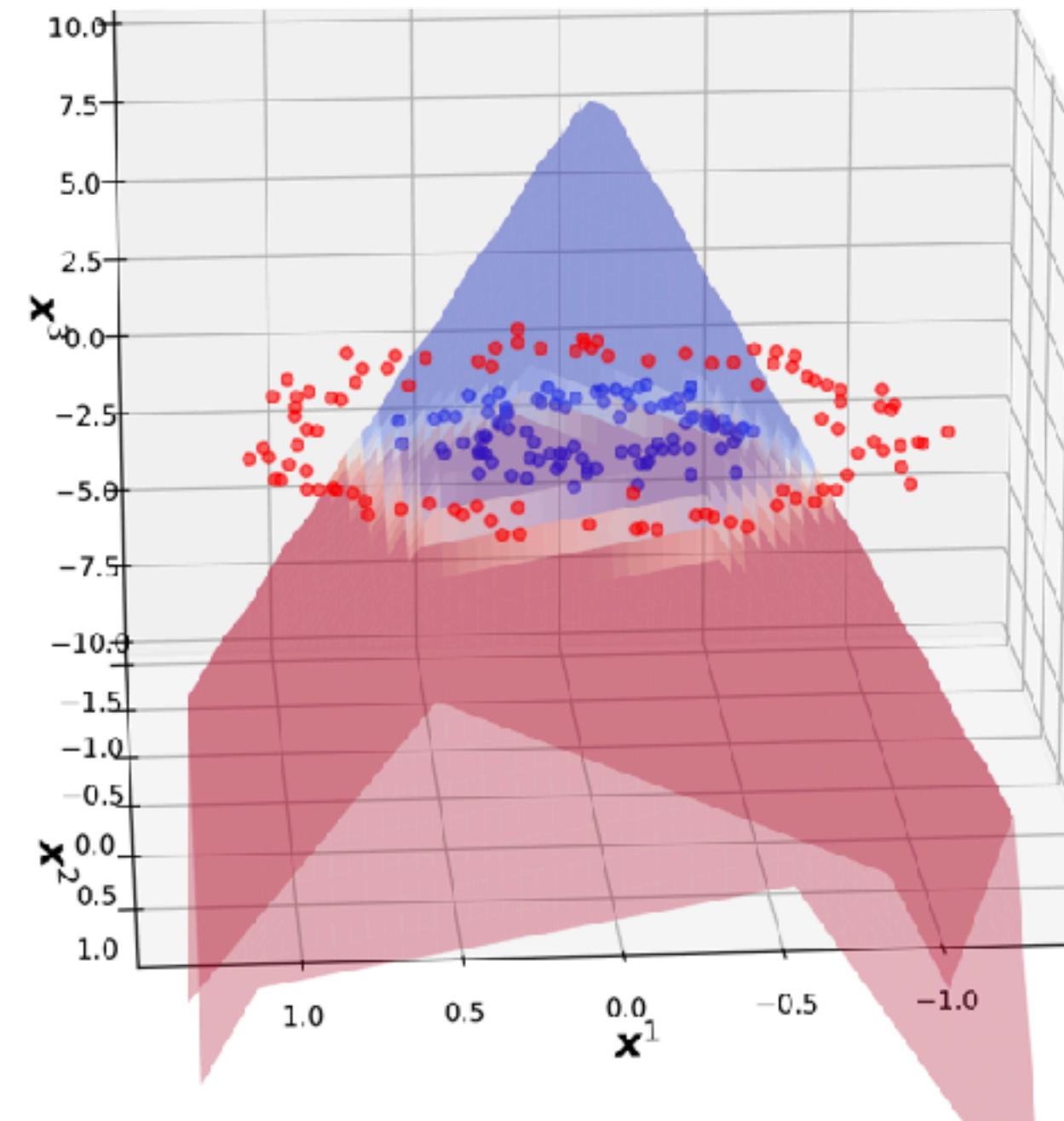
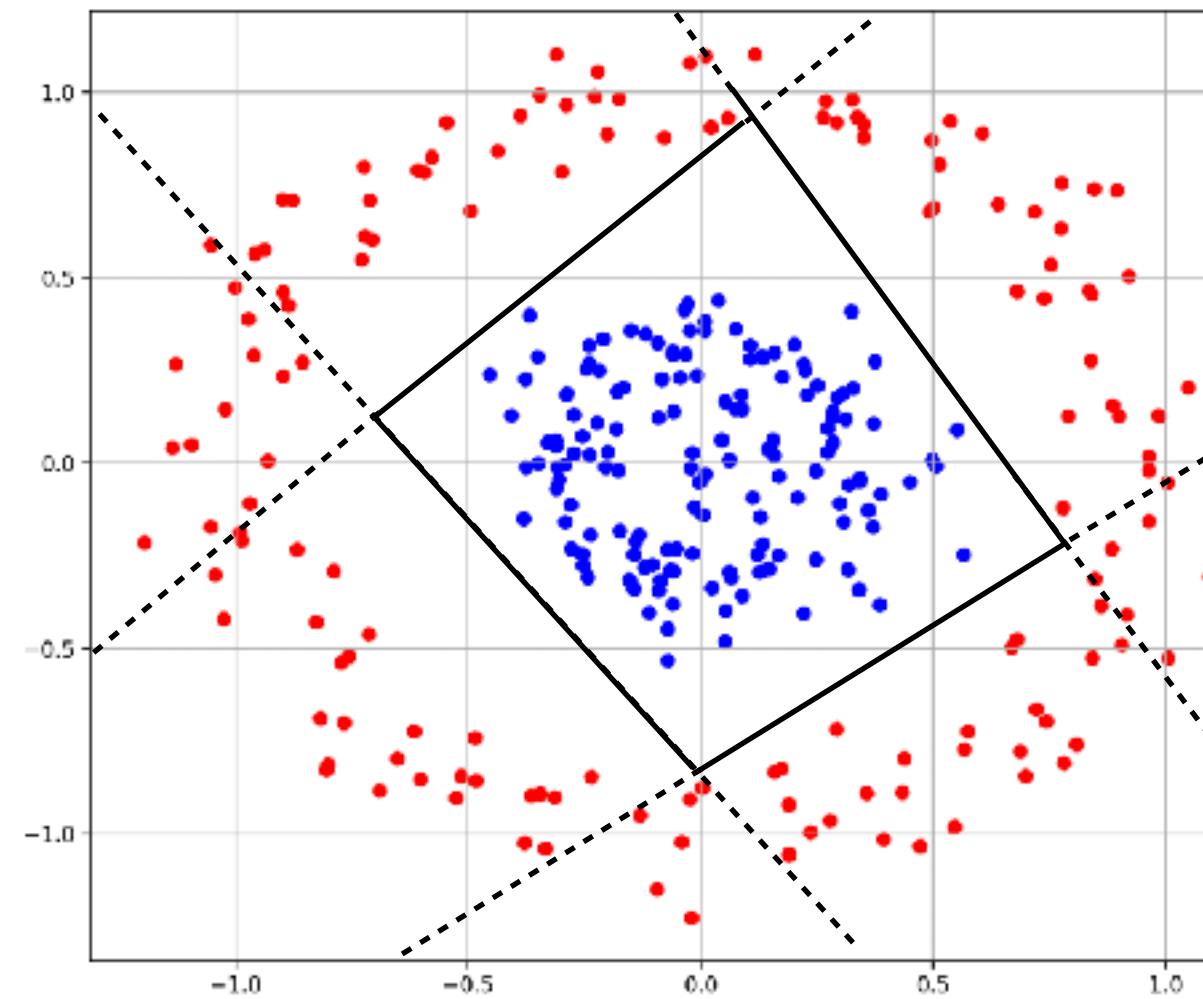
$$p_i \stackrel{\text{def}}{=} \mathbb{P}_{\alpha, \beta}(Y = 1 | \mathbf{x}_i) = \text{sigmoid}(\max(\mathbf{x}_i^\top \boldsymbol{\beta}^1 + \alpha_1, \mathbf{x}_i^\top \boldsymbol{\beta}^2 + \alpha_2))$$

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

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

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

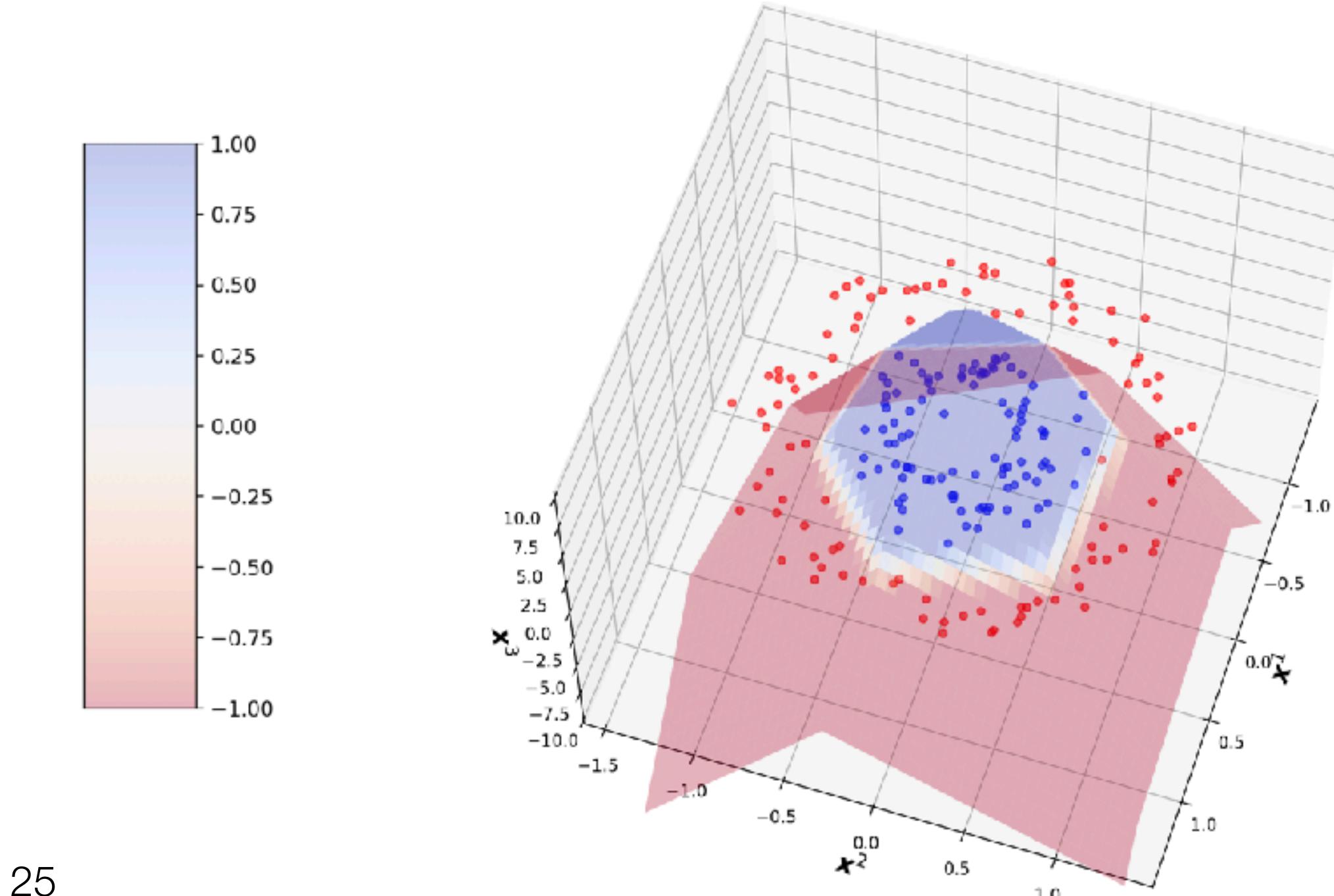
$$\vdots$$

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

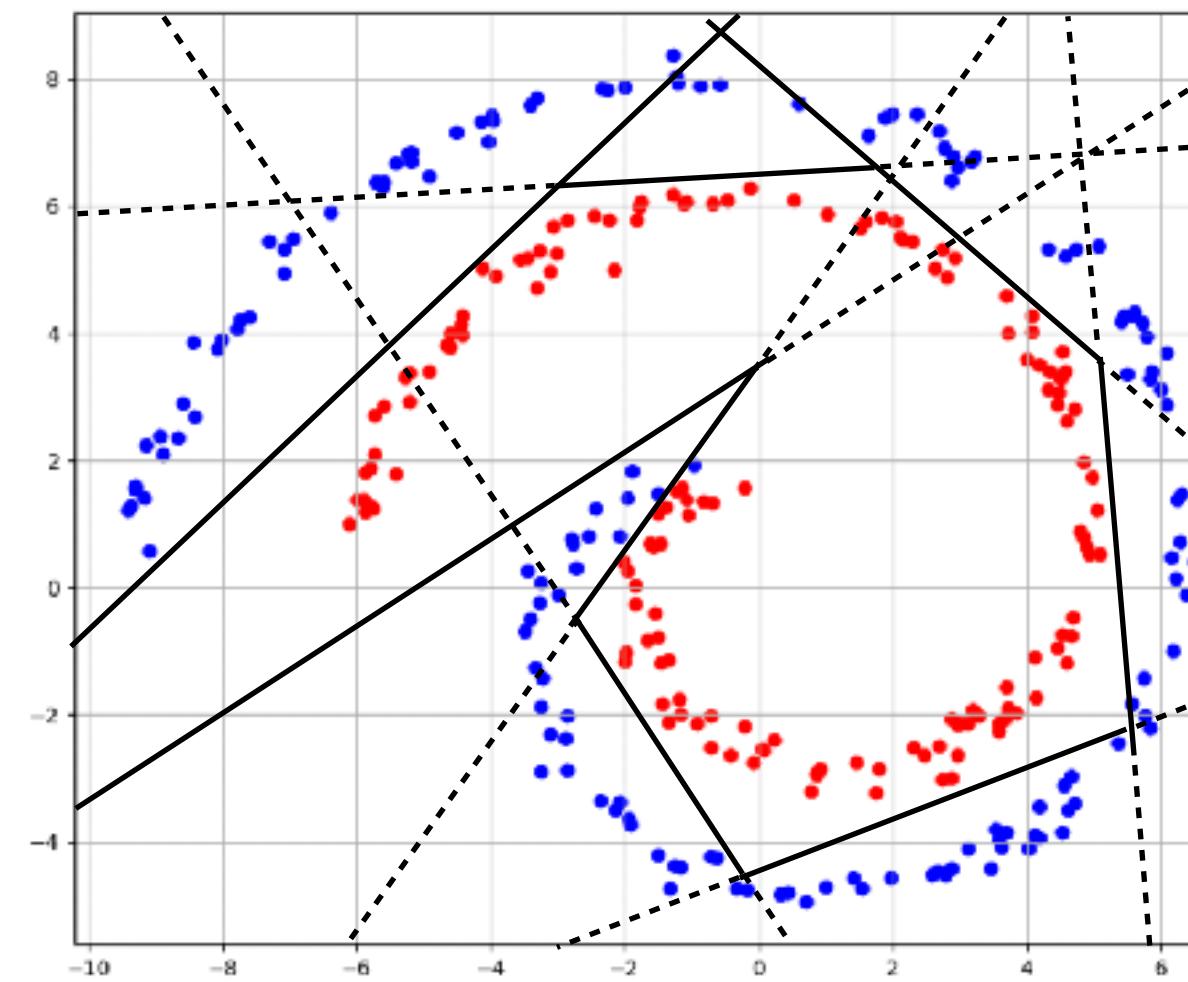
non-linéarité

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

sigmoid



Comment adapter ce modèle à des données plus complexes ?



Linéarités

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

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

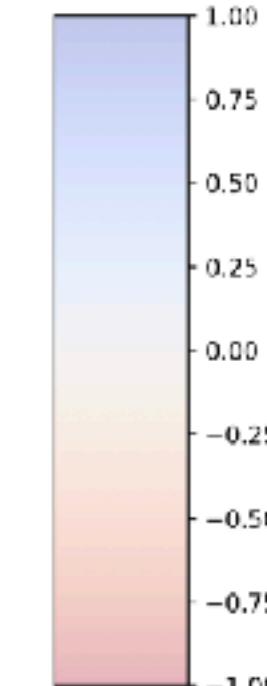
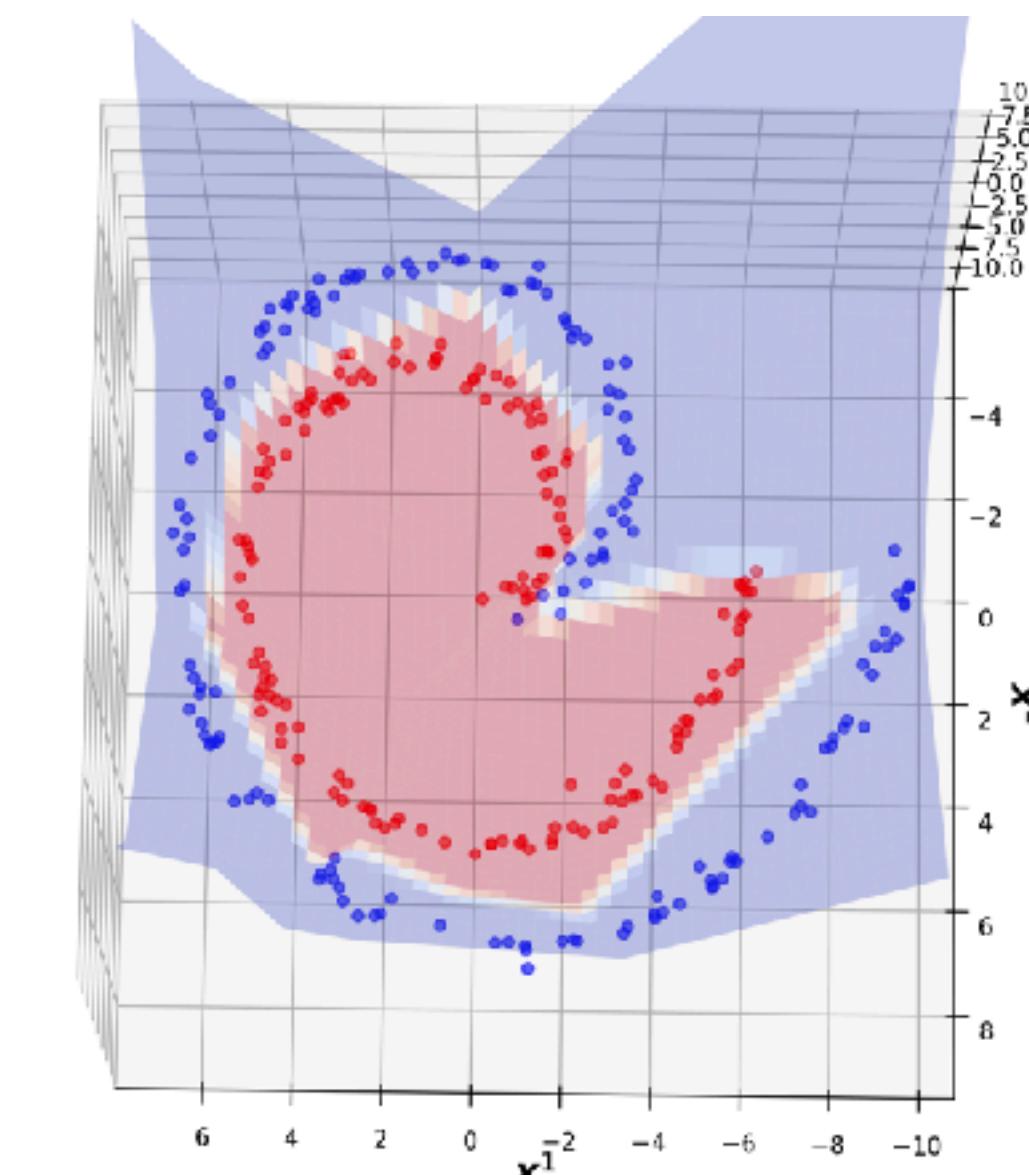
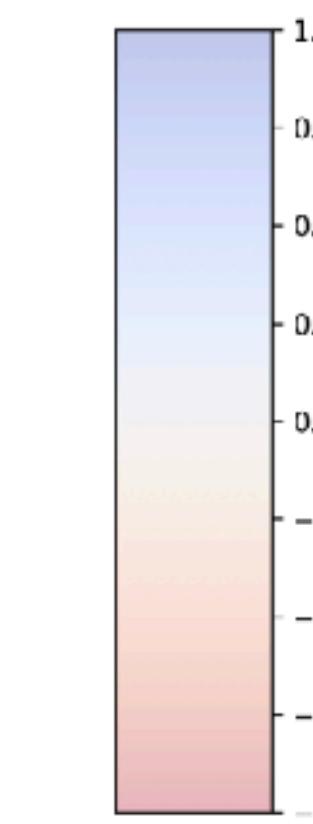
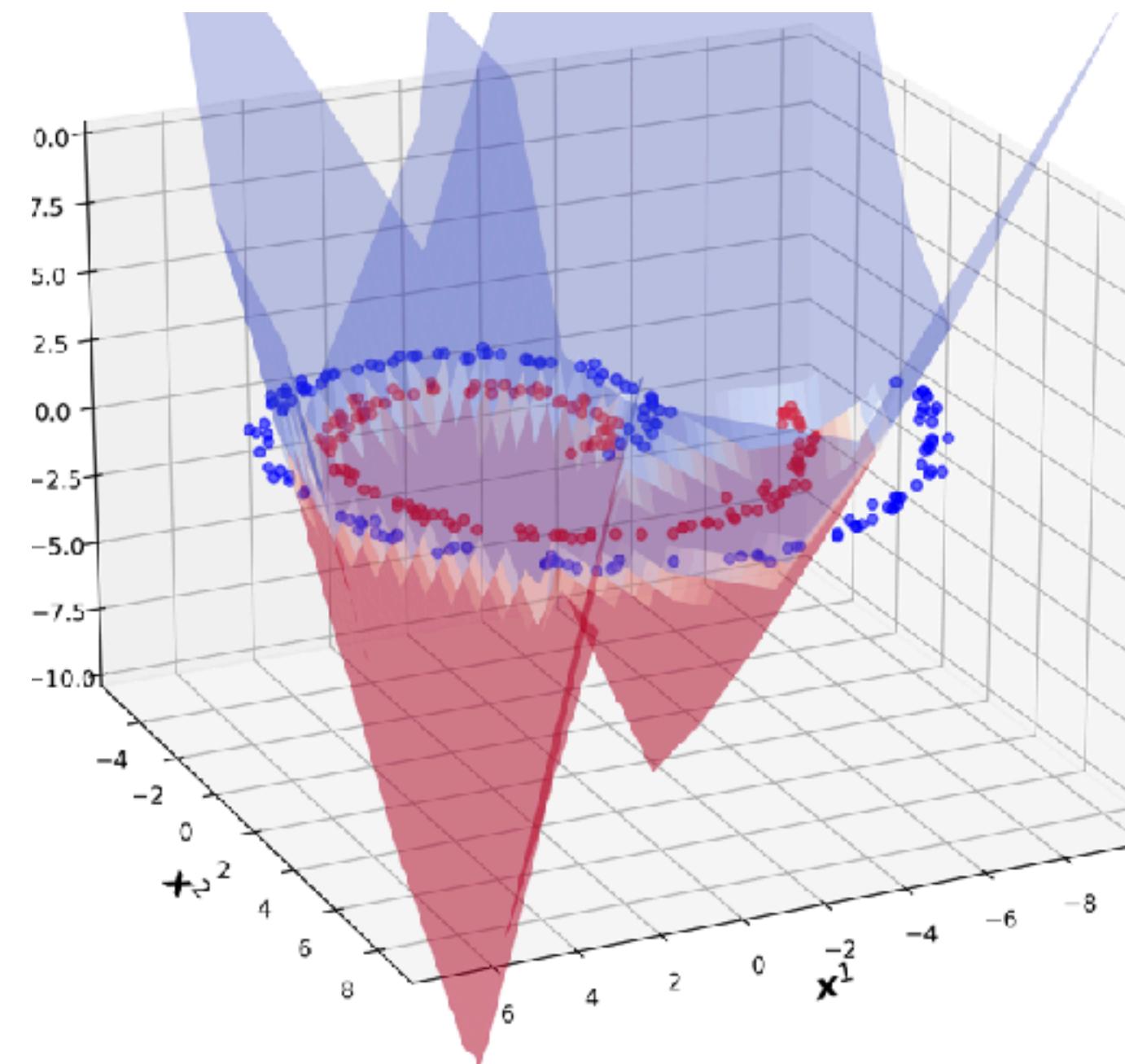
\vdots

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

non-linéarité

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

sigmoid

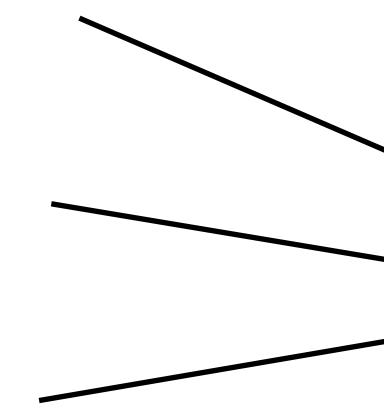


Linéarités

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

$$\vdots$$

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



non-linéarité

$$\max(z_1, \dots, z_p) \longrightarrow \text{sigmoid}$$

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

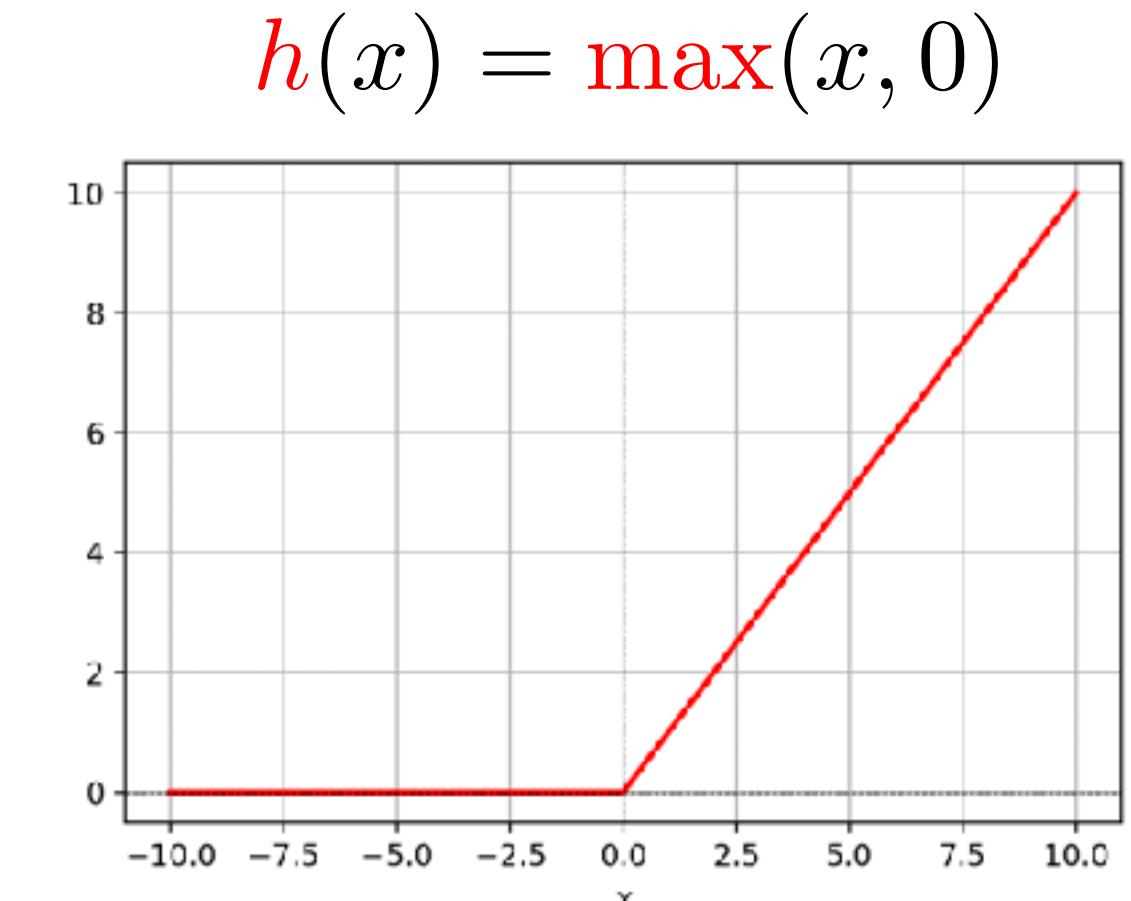
$$\begin{aligned}
 z_1 &\stackrel{\text{def}}{=} h(\mathbf{x}^\top \boldsymbol{\beta}^1 + \alpha_1) \\
 &\vdots \\
 z_p &\stackrel{\text{def}}{=} h(\mathbf{x}^\top \boldsymbol{\beta}^p + \alpha_p)
 \end{aligned}
 \longrightarrow \sum_{j=1}^p \omega_j z_j + \omega_0 \downarrow \text{sigmoid}$$



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

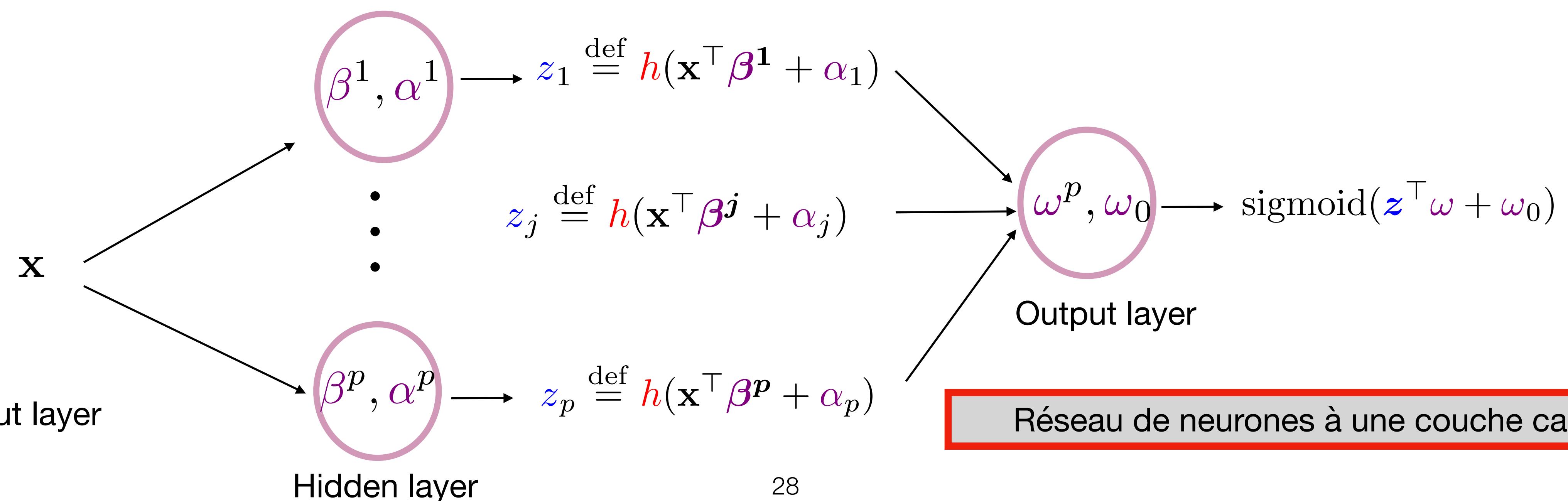
sigmoid($\sum_{j=1}^p \omega_j z_j + \omega_0$) = sigmoid($\mathbf{z}^\top \boldsymbol{\omega} + \omega_0$)

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

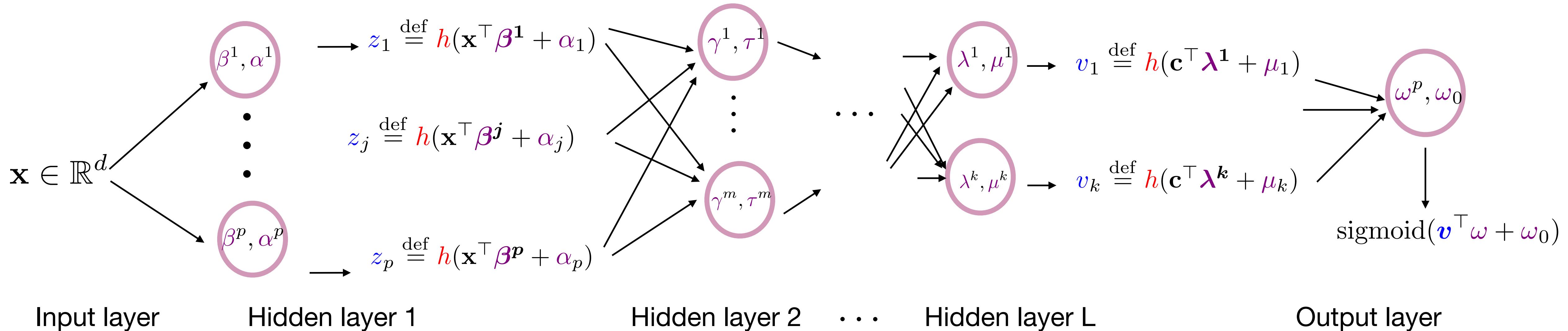


ReLU: Rectified Linear Unit

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...



Input layer

Hidden layer 1

Hidden layer 2

 \dots

Hidden layer L

Output layer

La dimension de chaque couche = le nombre de neurones

d



p



m



k



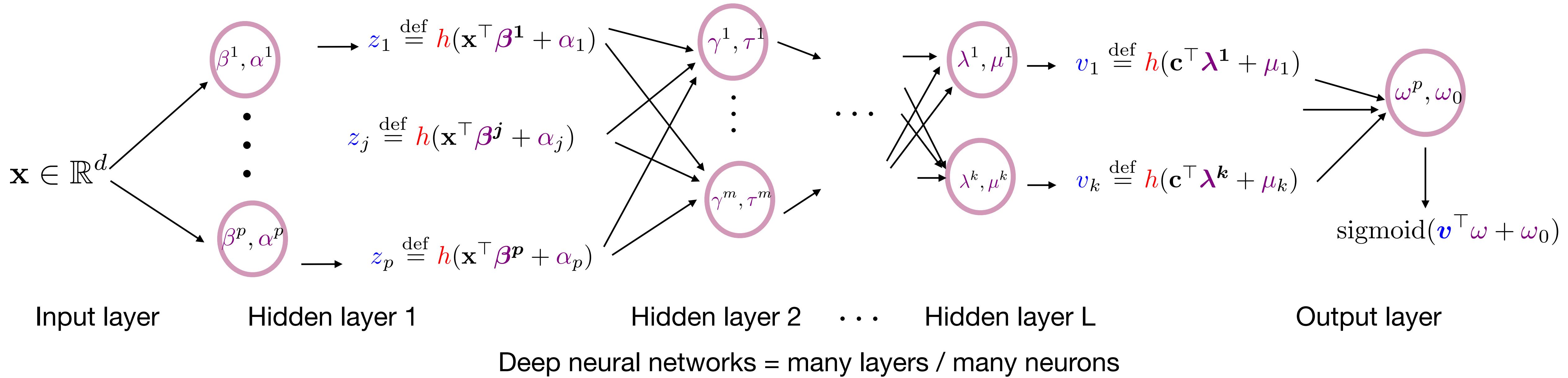
1

La profondeur du réseau = le nombre de layers

“Deep learning” = beaucoup de layers

la non-linéarité h est appelée: fonction d'activation

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



Comment peut-on définir l'output du Hidden layer 1 ($\mathbf{z} \in \mathbb{R}^p$) en une seule équation ?

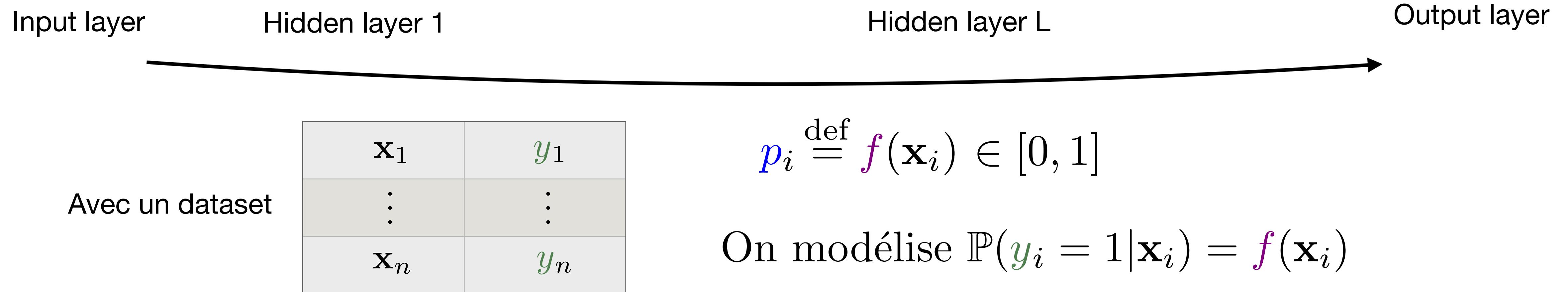
$$\begin{aligned} z_1 &= h(\beta_1^\top \mathbf{x} + \alpha_1) \\ &\vdots \\ z_p &= h(\beta_p^\top \mathbf{x} + \alpha_p) \end{aligned} \Leftrightarrow \mathbf{z} = h \left(\begin{bmatrix} - & \beta_1^\top & - \\ & \vdots & \\ - & \beta_p^\top & - \end{bmatrix} \begin{bmatrix} \mathbf{x} \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_p \end{bmatrix} \right) = h(\mathbf{B}\mathbf{x} + \boldsymbol{\alpha})$$

$\mathbf{B} \in \mathbb{R}^{p \times d}$
Weights

$\boldsymbol{\alpha} \in \mathbb{R}^p$
bias

h appliquée element-wise (numpy style)

$$\mathbf{x} \longrightarrow h(W_1 \mathbf{x} + \mathbf{b}_1) \rightarrow \mathbf{z}_1 \rightarrow \dots \rightarrow h(W_L \mathbf{z}_{L-1} + \mathbf{b}_L) \rightarrow \mathbf{z}_L \in \mathbb{R}^k \rightarrow \text{sigmoid}(w_{L+1}^\top \mathbf{z}_L + b_{L+1})$$



Comme avec la régression logistique, on “apprend” les paramètres en minimisant la cross-entropy loss:

$$\min_{\substack{W_1, \dots, W_{L+1} \\ b_1, \dots, b_{L+1}}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(p_i, y_i)$$

On suppose que les $y_i \in \mathbb{R}$ (Problème de régression) comment doit-on modifier le modèle ?

Jusqu'à présent nous avons considéré la classification binaire uniquement: deux classes (0 / 1) en modélisant $\mathbb{P}(\textcolor{teal}{y}_i = 1 | \mathbf{x}_i) = \textcolor{violet}{f}(\mathbf{x}_i)$

Nous avons défini $\textcolor{violet}{f}$ telle que son output soit dans [0, 1]

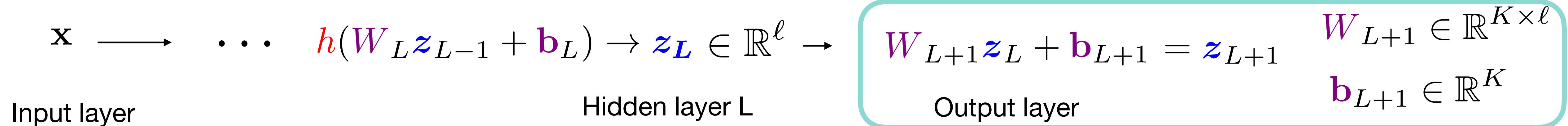
Comment généraliser pour $K > 2$ classes: $y_i \in \{0, 1, \dots, K - 1\}$?

Pour prédire la classe de \mathbf{x}_i , il faut calculer la probabilité de chaque classe: $\textcolor{blue}{p}_{i,k} = \mathbb{P}(\textcolor{teal}{y}_i = k | \mathbf{x}_i)$

Ainsi, il faut avoir un vecteur de probabilités $\mathbf{p}_i = \begin{bmatrix} \textcolor{blue}{p}_{i,0} \\ \vdots \\ \textcolor{blue}{p}_{i,K-1} \end{bmatrix}$
avec la contrainte $\sum_{k=0}^{K-1} \textcolor{blue}{p}_{i,k} = 1$

Comment peut-on modifier l'output layer tel que $\textcolor{violet}{f}(\mathbf{x})$ soit un vecteur dans $[0, 1]^K$ sommant à 1 ?

Pour que $f(x)$ soit un vecteur de dimension K , il faut avoir K neurones dans la dernière couche:



Comment peut-on transformer un vecteur $\mathbf{z} \in \mathbb{R}^K \rightarrow [0, 1]^K$ tel qu'il somme à 1 ?

$$\begin{bmatrix} z_1 \\ \vdots \\ z_K \end{bmatrix} \mapsto \mathbf{u} \stackrel{\text{def}}{=} \begin{bmatrix} \exp(z_1) \\ \vdots \\ \exp(z_K) \end{bmatrix} \in \mathbb{R}_+^K$$

Pour que \mathbf{u} somme à 1, il suffit de diviser par la somme des exponentielles:

$$\begin{bmatrix} z_1 \\ \vdots \\ z_K \end{bmatrix} \mapsto \mathbf{u} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\exp(z_1)}{\sum_{k=1}^K \exp(z_k)} \\ \vdots \\ \frac{\exp(z_K)}{\sum_{k=1}^K \exp(z_k)} \end{bmatrix} \in [0, 1]^K$$

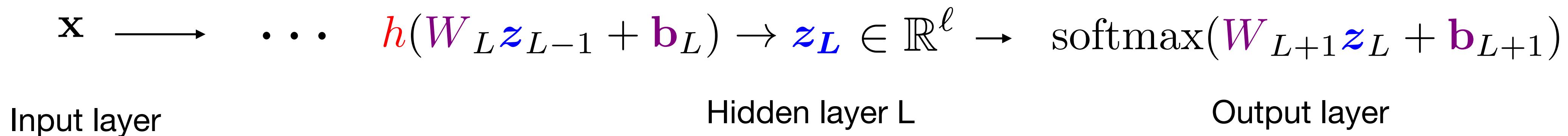
la fonction *softmax*

$$\mathbf{x} \longrightarrow \cdots h(W_L \mathbf{z}_{L-1} + \mathbf{b}_L) \rightarrow \mathbf{z}_L \in \mathbb{R}^\ell \rightarrow \text{softmax}(W_{L+1} \mathbf{z}_L + \mathbf{b}_{L+1})$$

Input layer

Hidden layer L

Output layer



On veut que $f(\mathbf{x}_i)$ corresponde au vecteur des probabilités $\mathbf{p}_i = \begin{bmatrix} \mathbb{P}(y_i = 0 | \mathbf{x}_i) \\ \vdots \\ \mathbb{P}(y_i = K-1 | \mathbf{x}_i) \end{bmatrix}$

Pour comparer les \mathbf{p}_i aux y_i , on transforme les labels en vecteurs de probabilités:

$$y = 0 \Rightarrow \mathbf{y} = [1, 0, \dots, 0]^\top$$

$$y = 1 \Rightarrow \mathbf{y} = [0, 1, \dots, 0]^\top$$

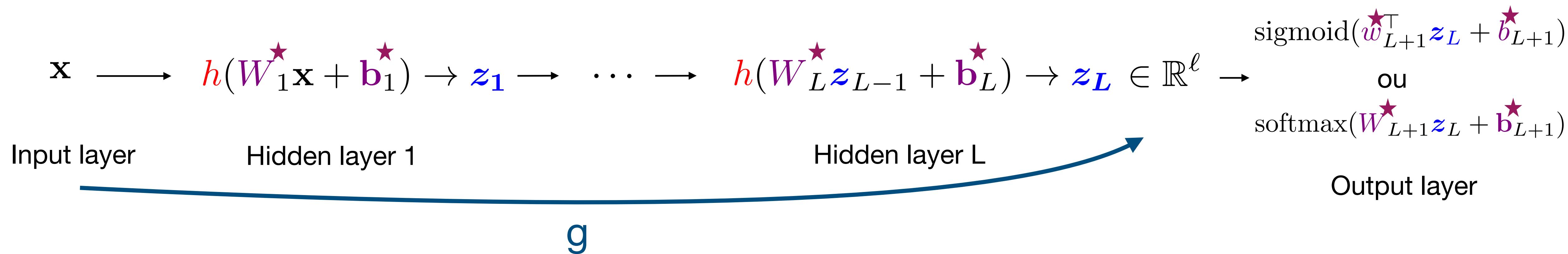
$$y = K-1 \Rightarrow \mathbf{y} = [0, 0, \dots, 1]^\top$$

$$\mathcal{L}(\mathbf{p}, \mathbf{y}) = \sum_{k=0}^{K-1} -\mathbf{y}_k \log(\mathbf{p}_k)$$

Cross-entropy

Vecteurs “one-hot”

On suppose que les poids sont optimisés et que le modèle a une excellente performance sur des données de test



On définit la transformation \mathbf{g} des données en s'arrêtant à au dernier hidden layer: $\mathbf{g} : \mathbb{R}^d \rightarrow \mathbb{R}^\ell \quad \ell < d$

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

Plus facile: car une simple régression logistique (output layer) a suffi pour les classifier: ils sont **forcément** linéairement séparables

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

L'output layer est souvent appelé “*classification head*”