

Applied QML

Lecture 8: Quantum Kernel Methods

Christophe Pere

2024-02-29

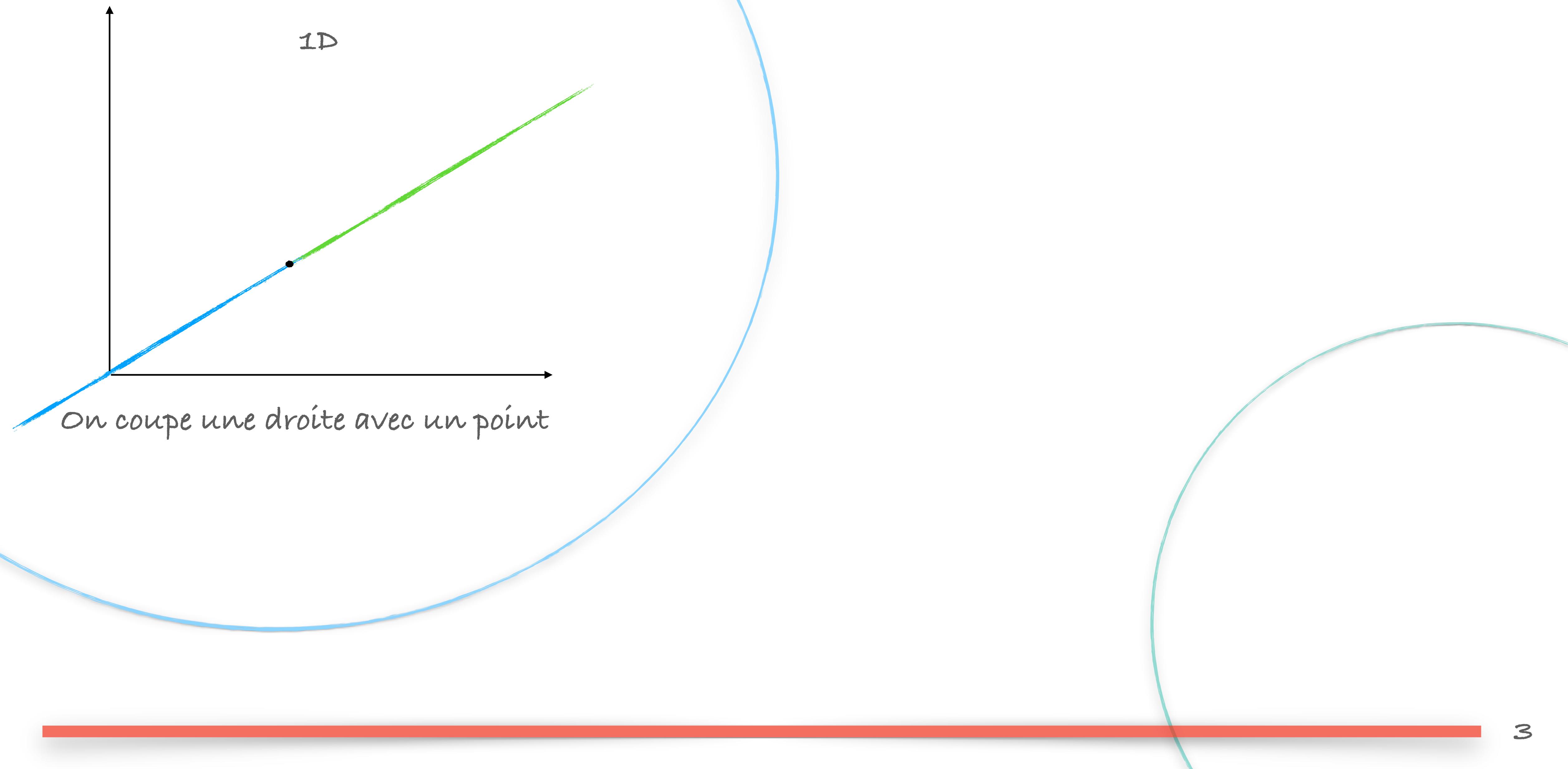
Table des matières

SVM

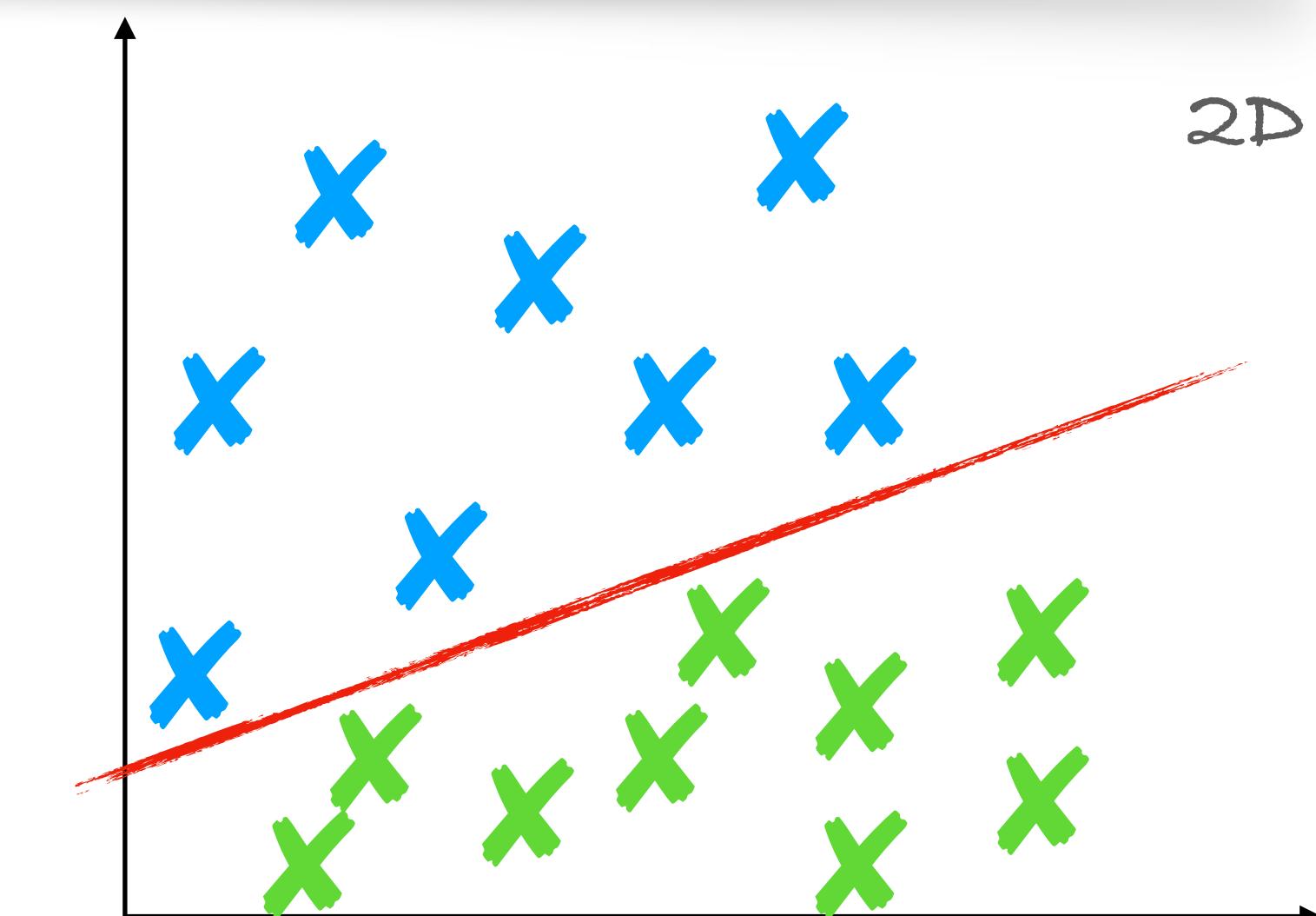
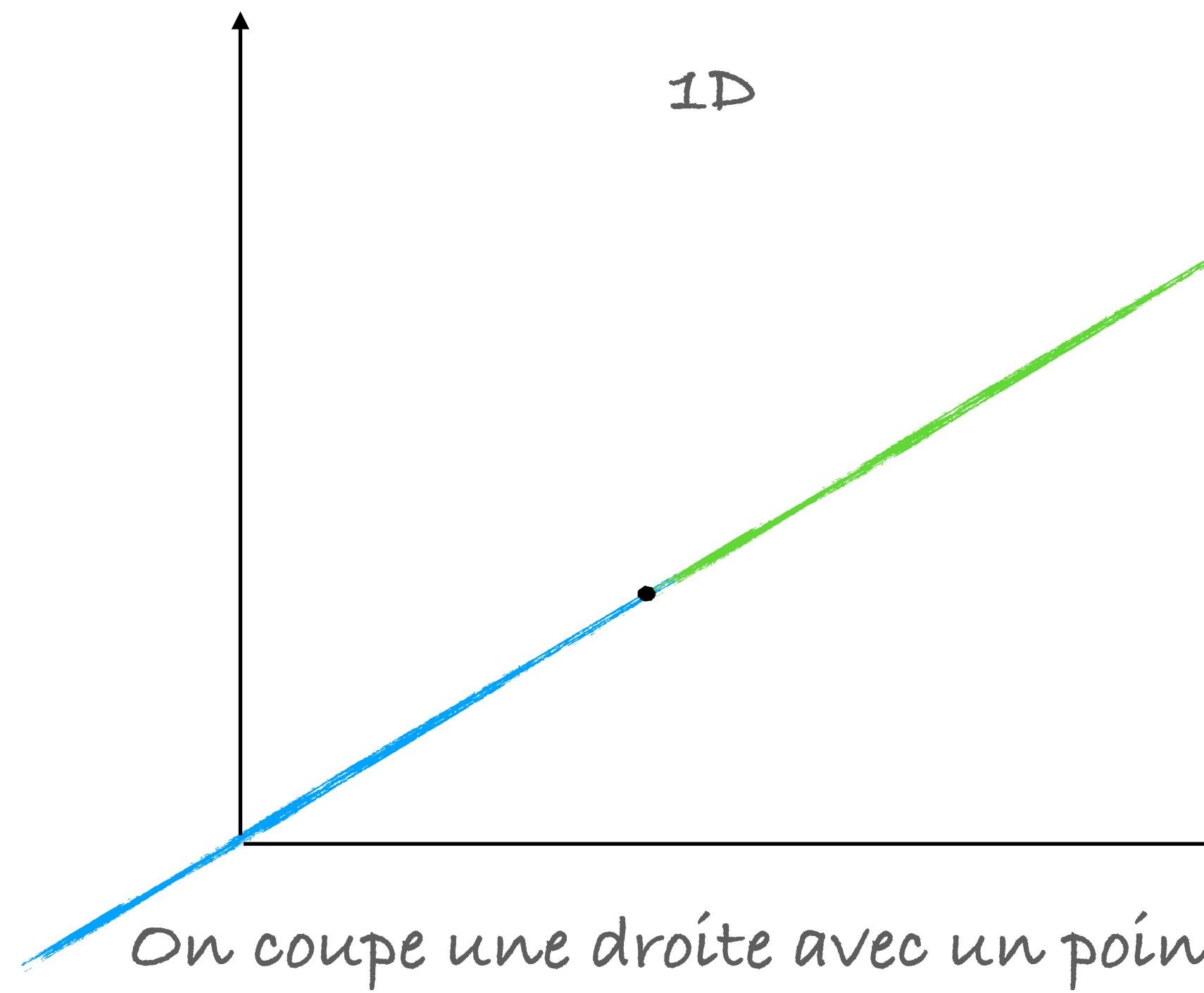
QSVM

Implementation

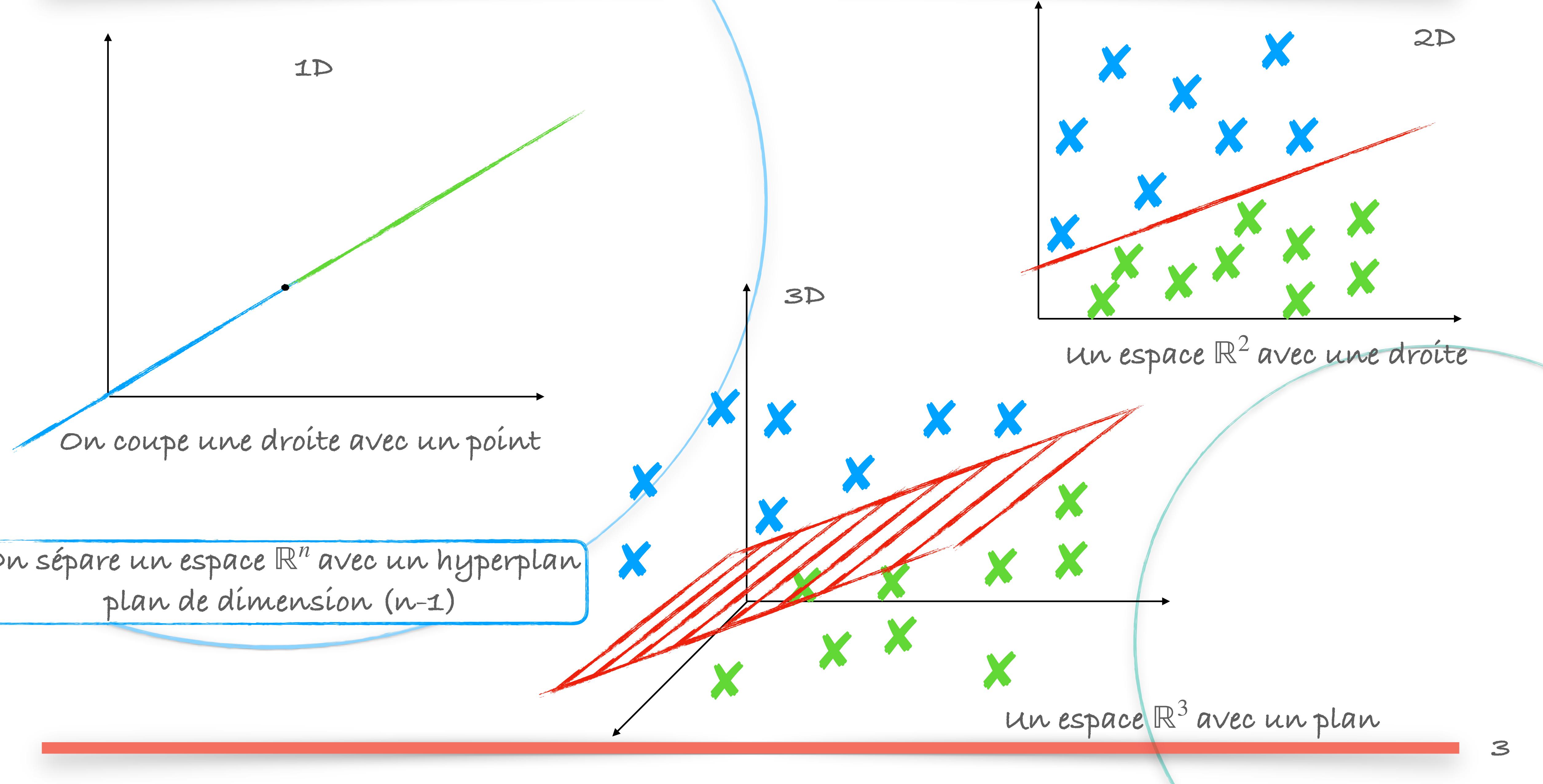
Support Vector Machines



Support Vector Machines



Support Vector Machines



Support Vector Machines

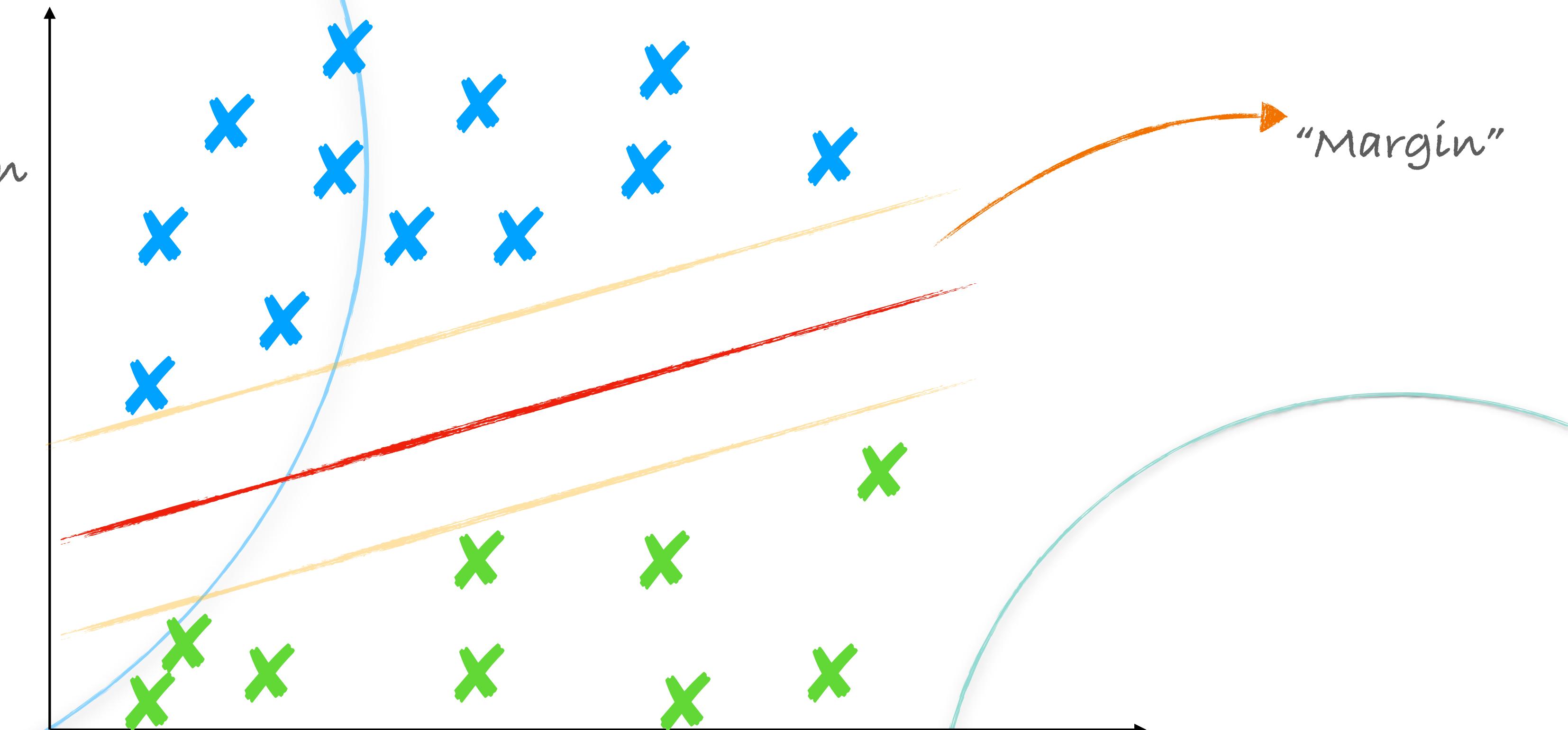
Maximisation de la zone de séparation
Problème d'optimisation

Ligne dans un plan est caractérisée
par $\vec{w} \in \mathbb{R}^2$ et un nombre réel $b \in \mathbb{R}$
avec les points $\vec{x} = (x_1, x_2)$

$$\vec{w} \cdot \vec{x} + b = 0$$

$$\vec{w} \cdot \vec{x} = w_1 x_1 + w_2 x_2$$

\vec{w} est la normale (perpendiculaire) à la ligne et b est l'intersection (l'ordonnée).



Support Vector Machines

Maximisation de la zone de séparation
Problème d'optimisation

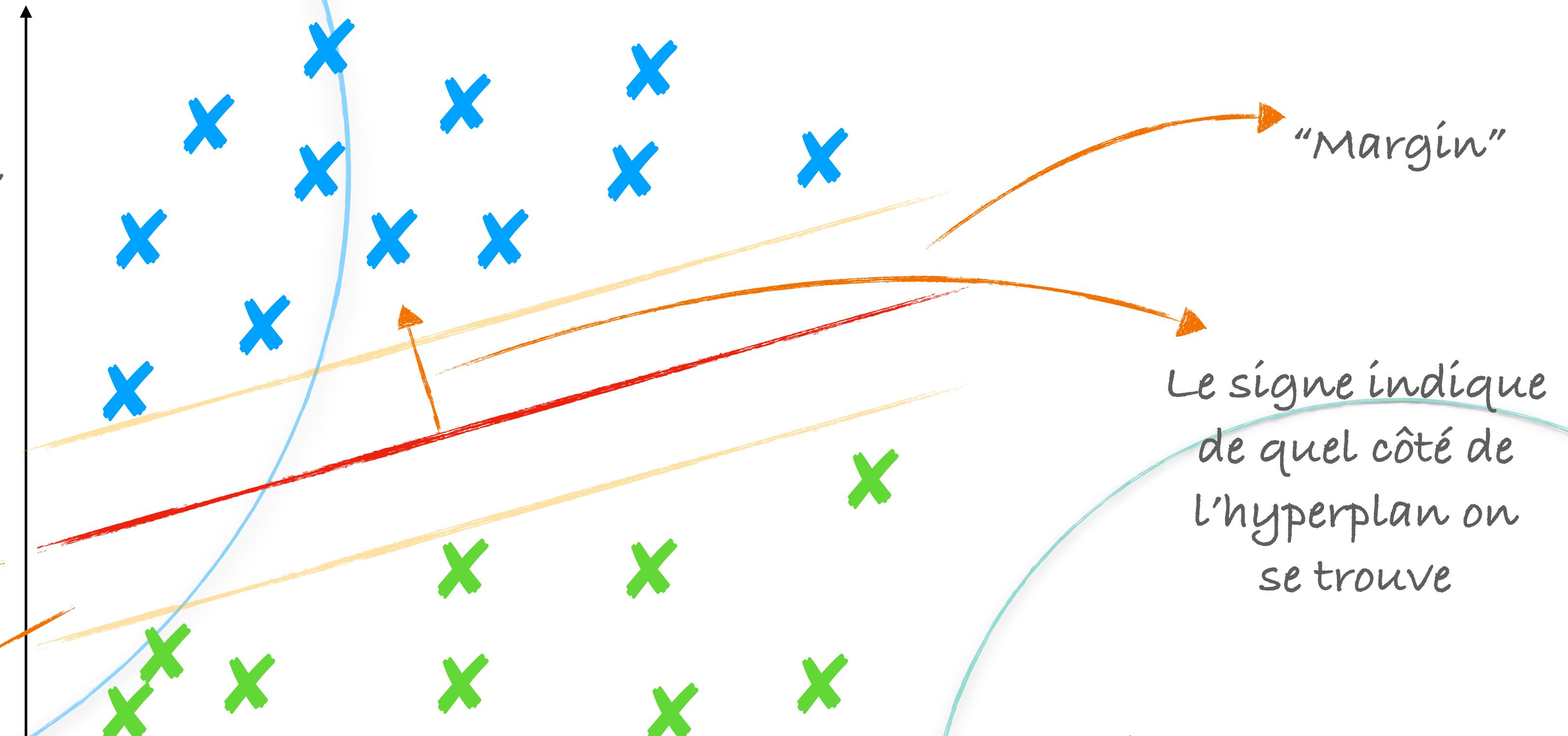
Ligne dans un plan est caractérisée
par $\vec{w} \in \mathbb{R}^2$ et un nombre réel $b \in \mathbb{R}$
avec les points $\vec{x} = (x_1, x_2)$

$$\vec{w} \cdot \vec{x} + b = 0$$

$$\vec{w} \cdot \vec{x} = w_1 x_1 + w_2 x_2$$

$$\vec{w} \cdot \vec{x} + b = C$$

\vec{w} est la normale (perpendiculaire) à la ligne et b est l'intersection (l'ordonnée).



Support Vector Machines

Minimiser $\|w\|$

Avec

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Pour m data

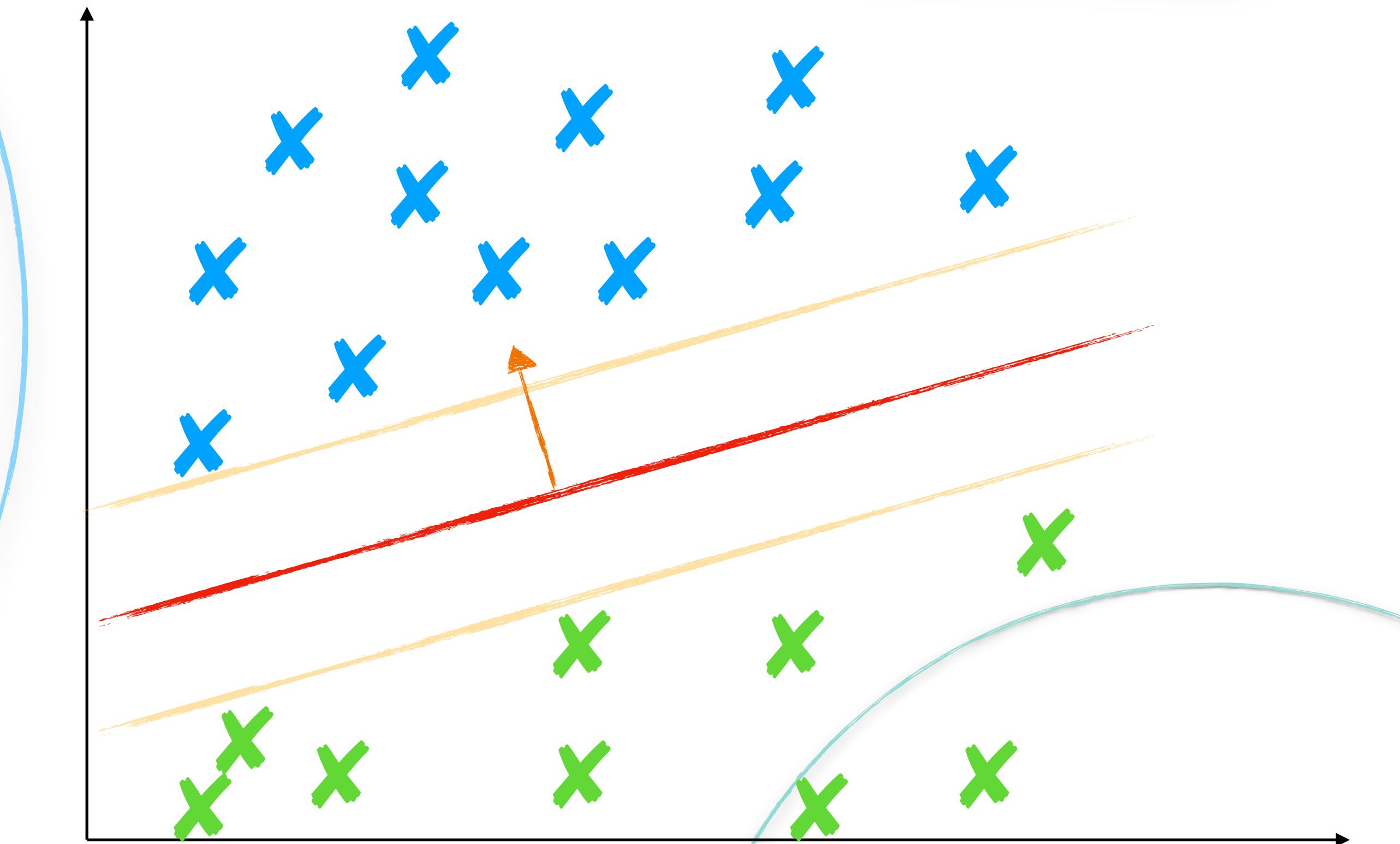
Hard-Margin

Minimiser $\frac{1}{2} \|w\|^2$

Avec

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$



Même problème mais plus efficace à minimiser car il n'y a plus de racine
implique que aucun élément ne peut être malclassifié ni présent dans la "margin"

Support Vector Machines

Minimiser $\|w\|$

Avec

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Pour m data

Hard-Margin

Minimiser $\frac{1}{2} \|w\|^2$

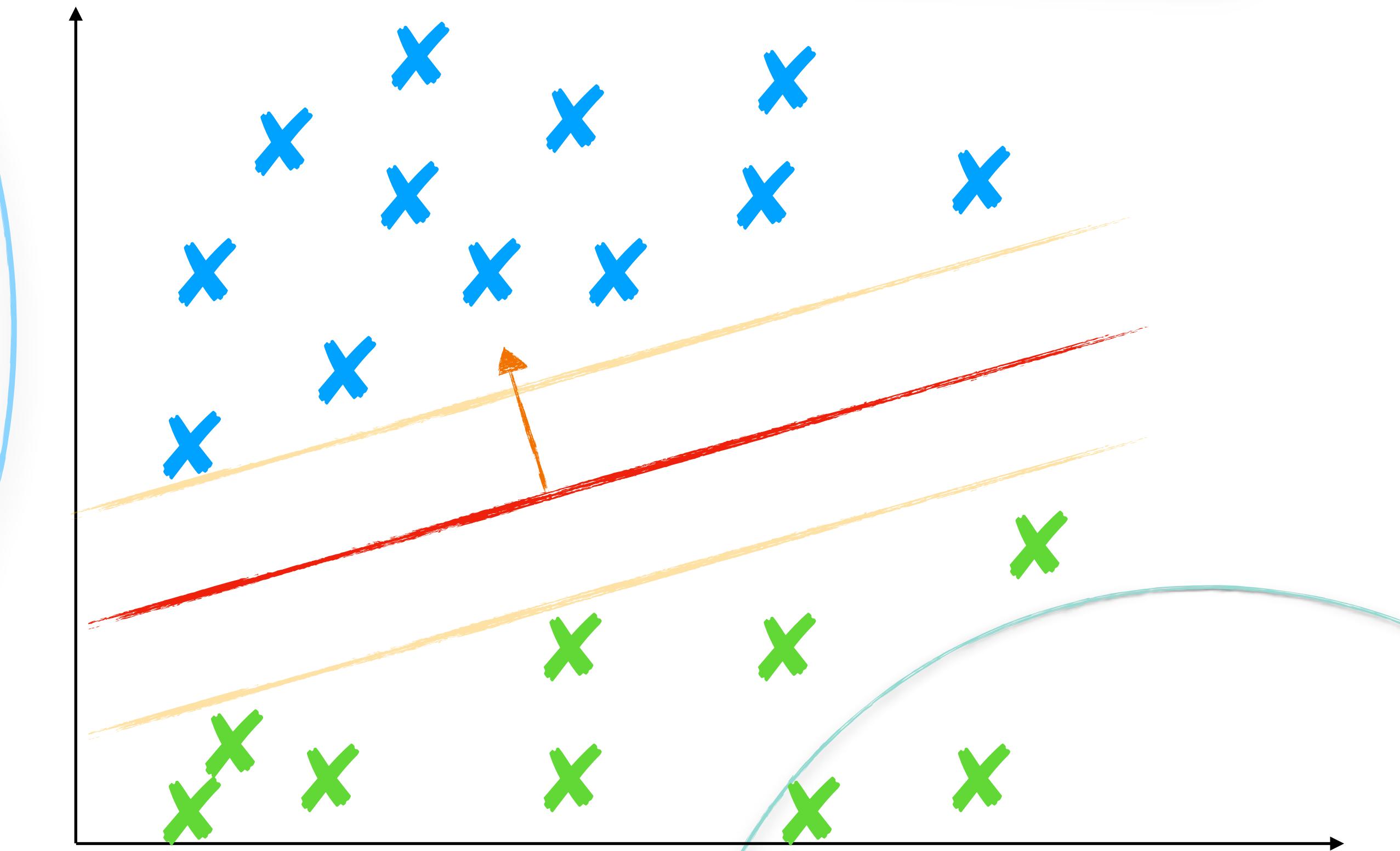
Avec

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Même problème mais plus efficace à minimiser car il n'y a plus de racine

implique que aucun élément ne peut être malclassifié ni présent dans la "margin"



Les données doivent donc être parfaitement séparables par un hyperplan

Support Vector Machines

Soft-margin

Hard-Margin

Minimiser $\frac{1}{2} \|w\|^2$

Avec $y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Introduction de "slack"
variables
Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Introduction de "slack"
variables
Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Si $\xi_j > 0$ le point peut-être proche
de l'hyperplan voire du mauvais côté

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser } \frac{1}{2} \|w\|^2$$

Avec $y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

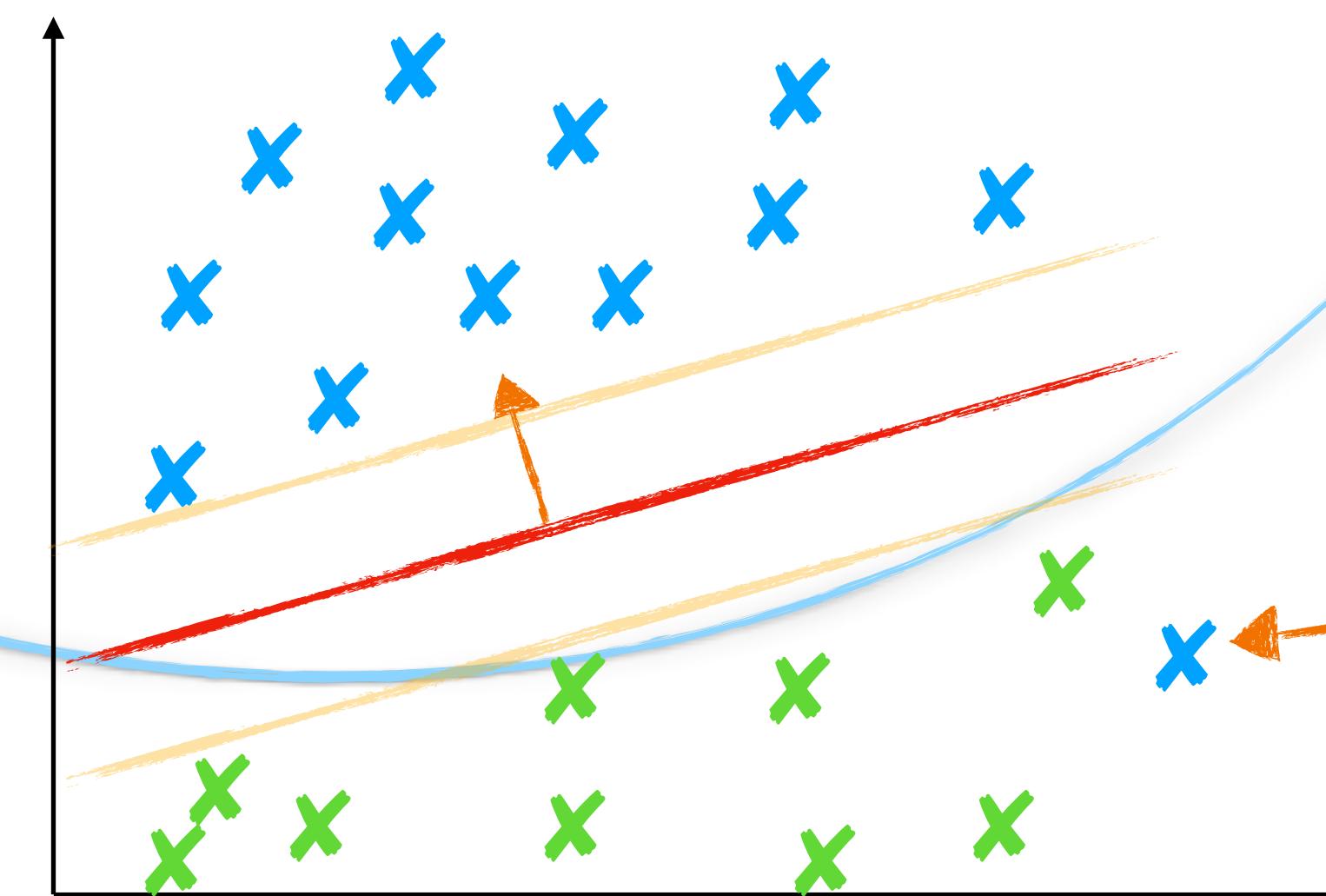
Introduction de "slack"
variables
Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser } \frac{1}{2} \|w\|^2$$

Avec $y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$



Si $\xi_j > 0$ le point peut-être proche
de l'hyperplan voire du mauvais côté

Plus gros ξ_j est et plus point sera du
mauvais côté de l'hyperplan

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

Avec $y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

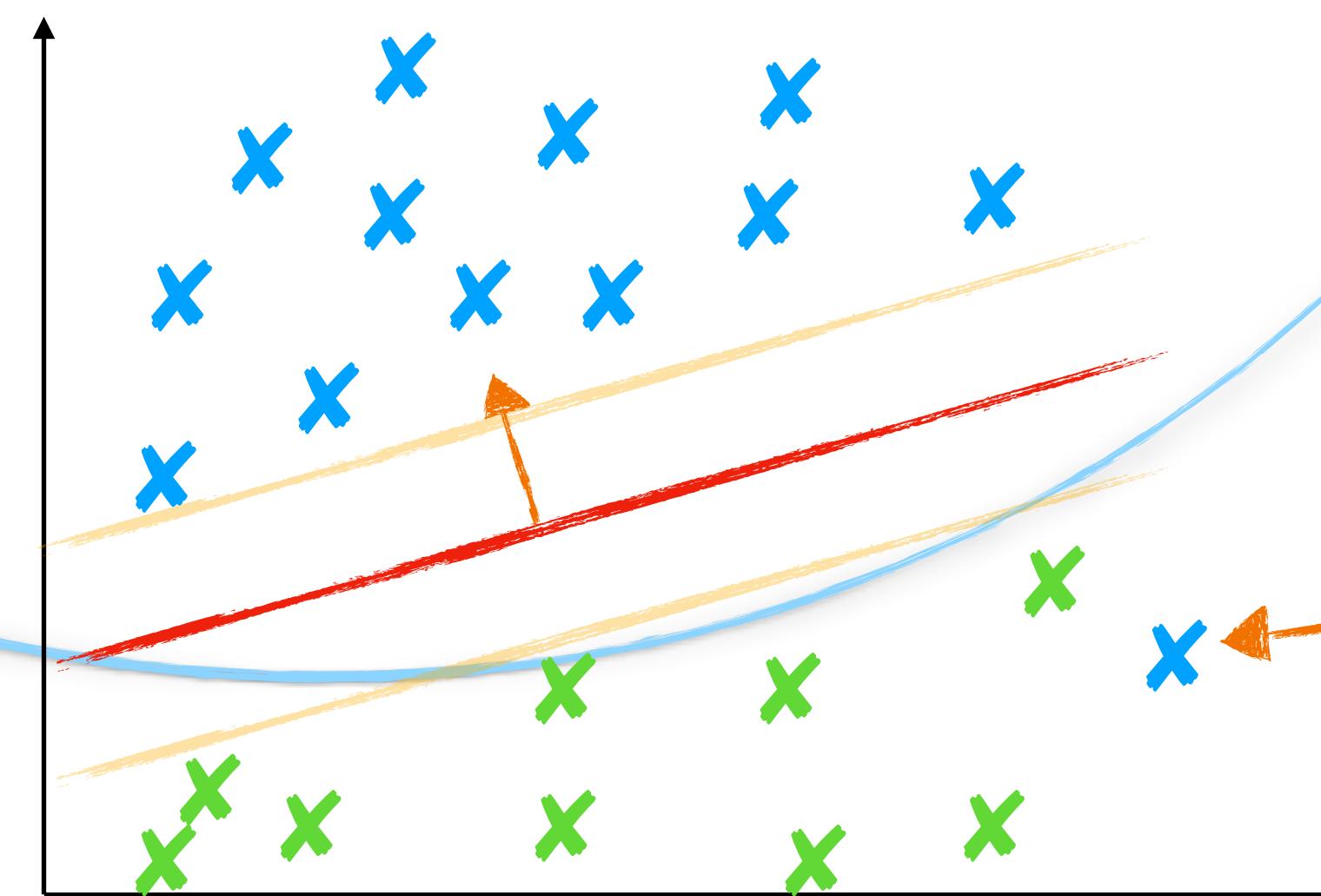
Introduction de "slack"
variables
Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

Avec

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$$
$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$



Si $\xi_j > 0$ le point peut-être proche de l'hyperplan voire du mauvais côté

Plus gros ξ_j est et plus point sera du mauvais côté de l'hyperplan

On veut que les ξ_j soient les plus petits possibles

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Introduction de "slack"
variables

Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2 + C \sum_j \xi_j$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

$$\xi_j \geq 0$$

Fonction de coût

C est un hyperparamètre de tolérance. Plus C est grand moins l'algorithme sera tolérant aux points dans la zone de "margin" et du mauvais côté de l'hyperplan.

Trouver la bonne valeur de C sans surapprendre
est la clef d'un bon apprentissage d'un SVM.

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|\vec{w}\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Introduction de "slack"
variables

Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser}$$

$$\frac{1}{2} \|\vec{w}\|^2 + C \sum_j \xi_j$$

$$\text{Avec}$$

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

$$\xi_j \geq 0$$

Fonction de coût pour entraîner un SVM

$$L(\vec{w}, b; \vec{x}, y) = \max \{0, 1 - y(\vec{w} \cdot \vec{x} + b)\}$$

Hinge-Loss

Revient à minimiser les éléments mal classifiés comme le terme ξ_j

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Introduction de "slack"
variables
Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser}$$

$$\frac{1}{2} \|w\|^2 + C \sum_j \xi_j$$

$$\text{Avec}$$

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

$$\xi_j \geq 0$$

En pratique le problème équivalent est le "Lagrangian dual"

$$\text{Minimiser} \quad \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} y_i y_k \alpha_j \alpha_k (\vec{x}_j \cdot \vec{x}_k)$$

$$\text{Avec}$$

$$0 \leq \alpha_j \leq C$$

$$\sum_j \alpha_j y_j = 0$$

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Introduction de "slack"
variables
Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser}$$

$$\frac{1}{2} \|w\|^2 + C \sum_j \xi_j$$

$$\text{Avec}$$

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

$$\xi_j \geq 0$$

En pratique le problème équivalent est le "Lagrangian dual"

On peut passer de α à w

$$\text{Minimiser}$$

$$\sum_j \alpha_j - \frac{1}{2} \sum_{j,k} y_i y_k \alpha_j \alpha_k (\vec{x}_j \cdot \vec{x}_k)$$

$$\text{Avec}$$

$$0 \leq \alpha_j \leq C$$

$$\sum_j \alpha_j y_j = 0$$

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Introduction de "slack"
variables
Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser}$$

$$\frac{1}{2} \|w\|^2 + C \sum_j \xi_j$$

$$\text{Avec}$$

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

$$\xi_j \geq 0$$

En pratique le problème équivalent est le "Lagrangian dual"

$$\vec{w} = \sum_j \alpha_j y_j \vec{x}_j$$

On peut passer de α à w

$$\text{Minimiser}$$

$$\sum_j \alpha_j - \frac{1}{2} \sum_{j,k} y_i y_k \alpha_j \alpha_k (\vec{x}_j \cdot \vec{x}_k)$$

$$\text{Avec}$$

$$0 \leq \alpha_j \leq C$$

$$\sum_j \alpha_j y_j = 0$$

Fonction de coût

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Introduction de "slack"
variables
Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser}$$

$$\frac{1}{2} \|w\|^2 + C \sum_j \xi_j$$

$$\text{Avec}$$

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

$$\xi_j \geq 0$$

En pratique le problème équivalent est le "Lagrangian dual"

Les w ne sont dépendant que des points du dataset x et sont appelés des ... support vectors

$$\vec{w} = \sum_j \alpha_j y_j \vec{x}_j$$

On peut passer de α à w

$$\text{Minimiser}$$

$$\sum_j \alpha_j - \frac{1}{2} \sum_{j,k} y_i y_k \alpha_j \alpha_k (\vec{x}_j \cdot \vec{x}_k)$$

$$\text{Avec}$$

$$0 \leq \alpha_j \leq C$$

$$\sum_j \alpha_j y_j = 0$$

Support Vector Machines

Soft-margin

Hard-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2$$

$$\text{Avec} \quad y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

Introduction de "slack"
variables
Appelé aussi tolérance

Soft-Margin

$$\text{Minimiser} \quad \frac{1}{2} \|w\|^2 + C \sum_j \xi_j$$

$$\text{Avec}$$

$$y_j(\vec{w} \cdot \vec{x}_j + b) \geq 1 - \xi_j$$

$$y_j \in \{1, -1\}, \quad j = 0, \dots, m$$

$$\xi_j \geq 0$$

Fonction de coût

Les données avec ces approches ne sont séparable que linéairement

Les w ne sont dépendant que des points du dataset x et sont appelés des ... support vectors

$$\vec{w} = \sum_j \alpha_j y_j \vec{x}_j$$

On peut passer de α à w

$$\text{Minimiser}$$

$$\text{Avec}$$

$$\sum_j \alpha_j - \frac{1}{2} \sum_{j,k} y_i y_k \alpha_j \alpha_k (\vec{x}_j \cdot \vec{x}_k)$$

$$0 \leq \alpha_j \leq C$$

$$\sum_j \alpha_j y_j = 0$$

Support Vector Machines

Kernel-Trick

L'objectif de l'approche est de "mapper" une donnée d'une dimension \mathbb{R}^n dans une plus haute dimension \mathbb{R}^N en espérant que cette projection soit séparable par un hyperplan

Support Vector Machines

Kernel-Trick

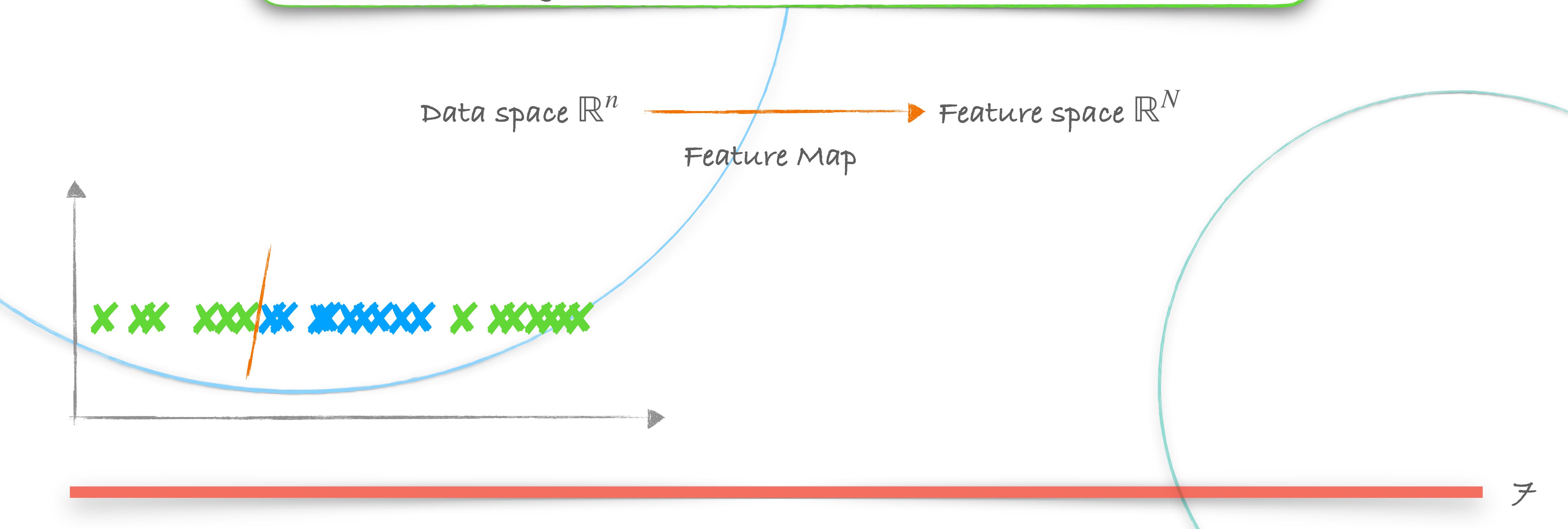
L'objectif de l'approche est de "mapper" une donnée d'une dimension \mathbb{R}^n dans une plus haute dimension \mathbb{R}^N en espérant que cette projection soit séparable par un hyperplan



Support Vector Machines

Kernel-Trick

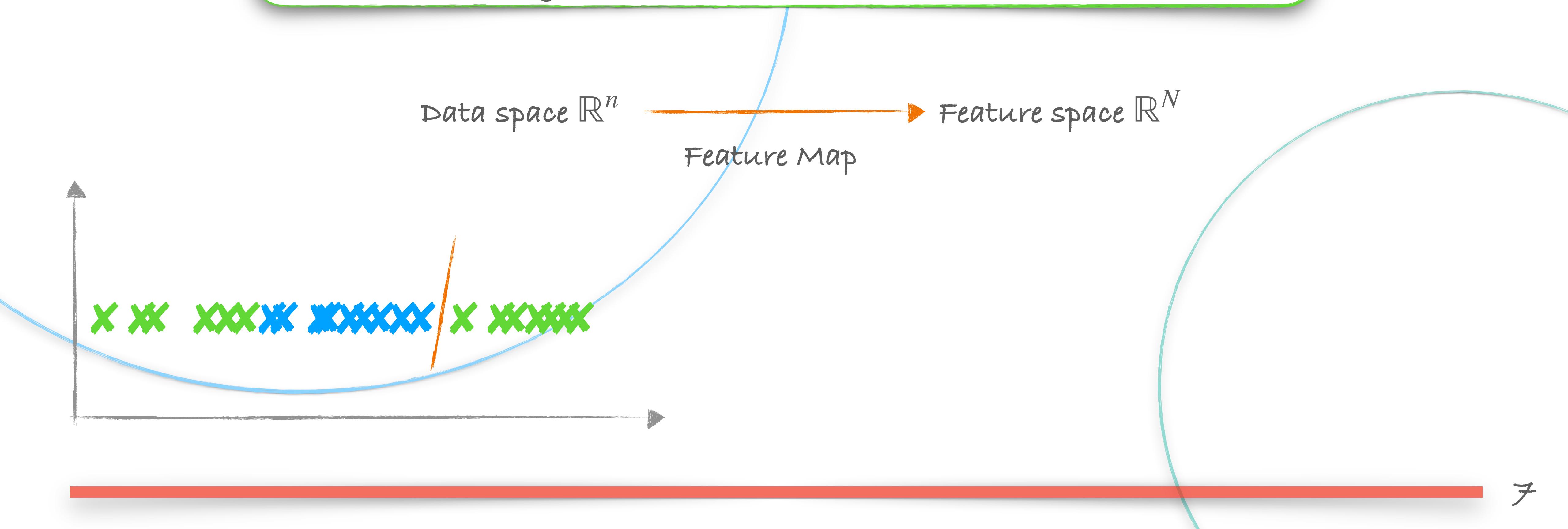
L'objectif de l'approche est de "mapper" une donnée d'une dimension \mathbb{R}^n dans une plus haute dimension \mathbb{R}^N en espérant que cette projection soit séparable par un hyperplan



Support Vector Machines

Kernel-Trick

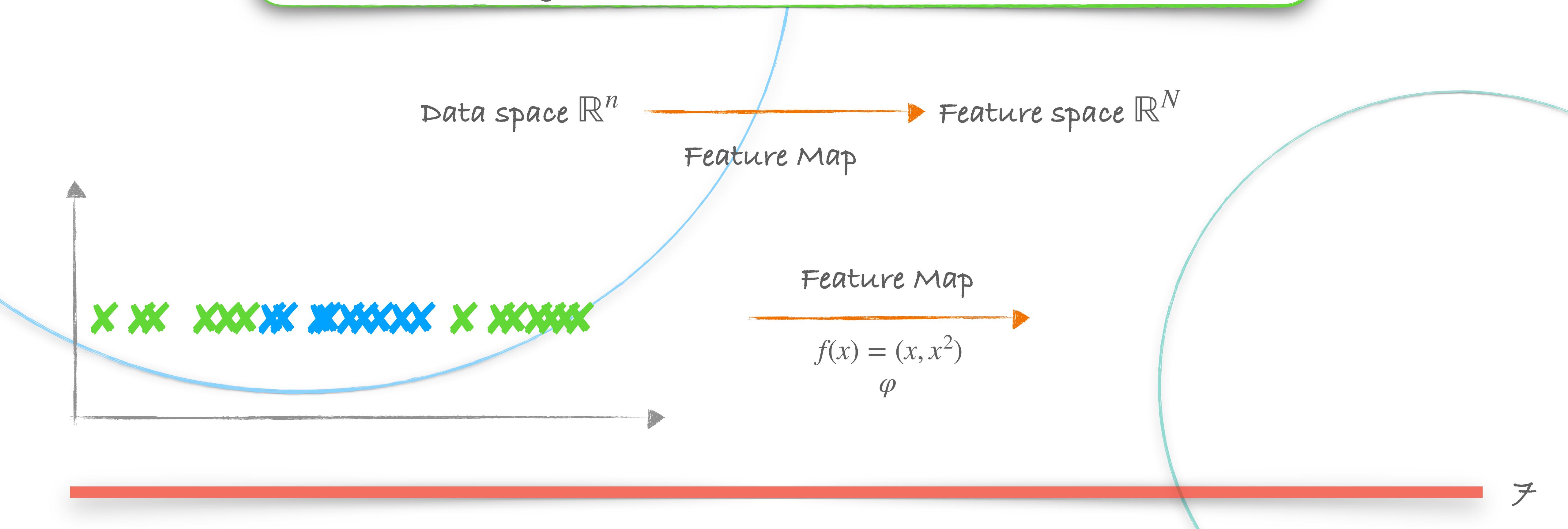
L'objectif de l'approche est de "mapper" une donnée d'une dimension \mathbb{R}^n dans une plus haute dimension \mathbb{R}^N en espérant que cette projection soit séparable par un hyperplan



Support Vector Machines

Kernel-Trick

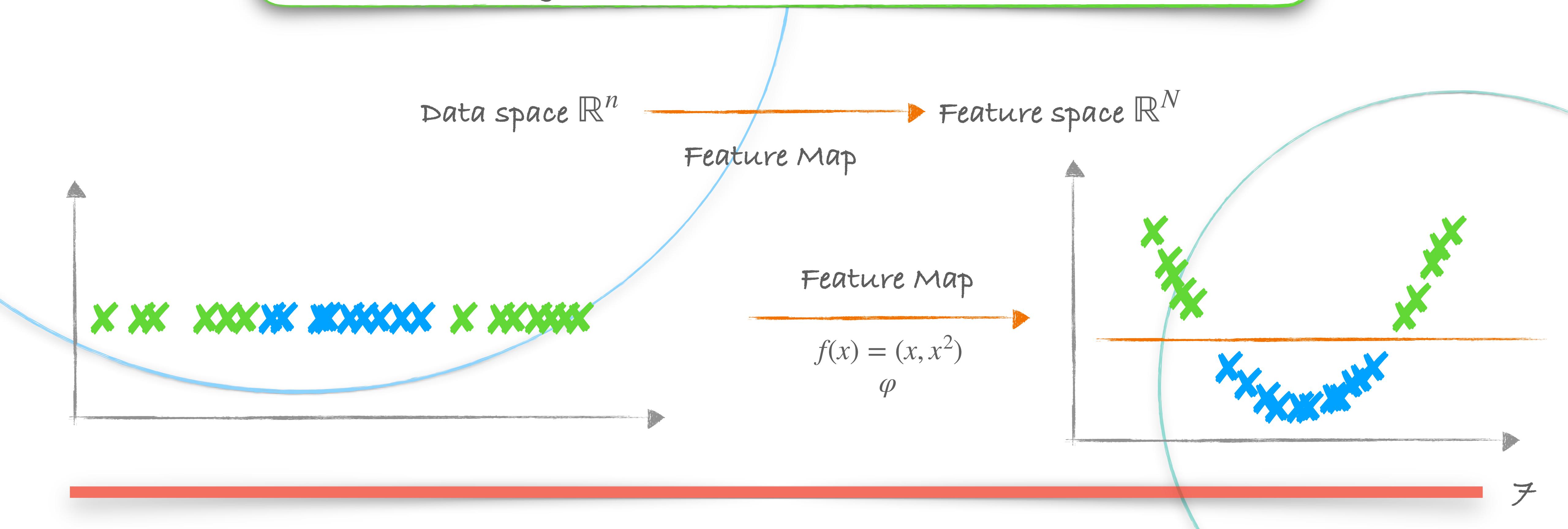
L'objectif de l'approche est de "mapper" une donnée d'une dimension \mathbb{R}^n dans une plus haute dimension \mathbb{R}^N en espérant que cette projection soit séparable par un hyperplan



Support Vector Machines

Kernel-Trick

L'objectif de l'approche est de "mapper" une donnée d'une dimension \mathbb{R}^n dans une plus haute dimension \mathbb{R}^N en espérant que cette projection soit séparable par un hyperplan



Support Vector Machines

Kernel-Trick

L'objectif de l'approche est de "mapper" une donnée d'une dimension \mathbb{R}^n dans une plus haute dimension \mathbb{R}^N en espérant que cette projection soit séparable par un hyperplan

Data space \mathbb{R}^n

Feature space \mathbb{R}^N

Feature Map

Feature Map

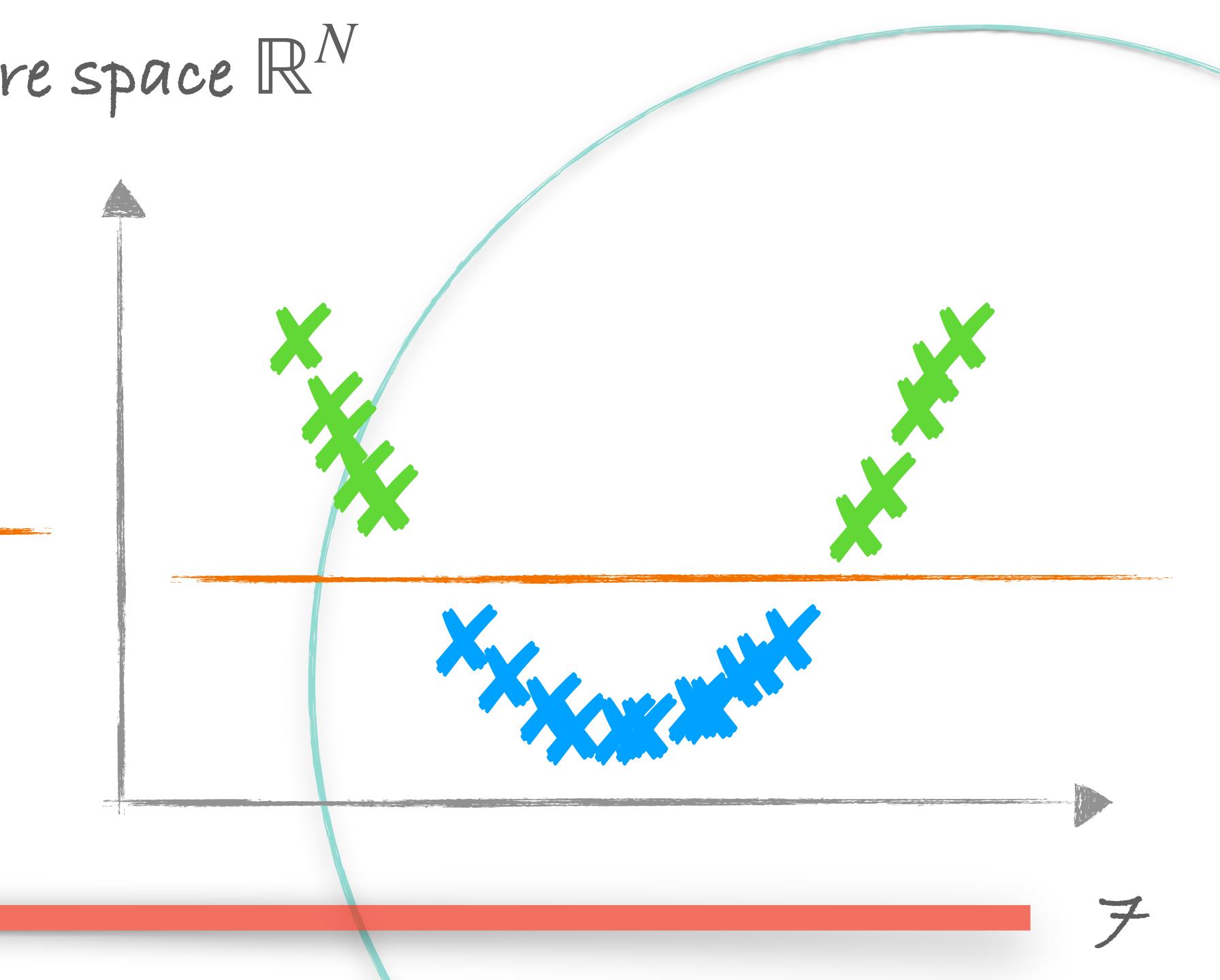
φ

L'opération qu'on doit effectuer dans le feature space est le inner product

Ce qui se traduit par un kernel!

$$k(x_1, x_2) = \varphi(\vec{x}_1) \cdot \varphi(\vec{x}_2)$$

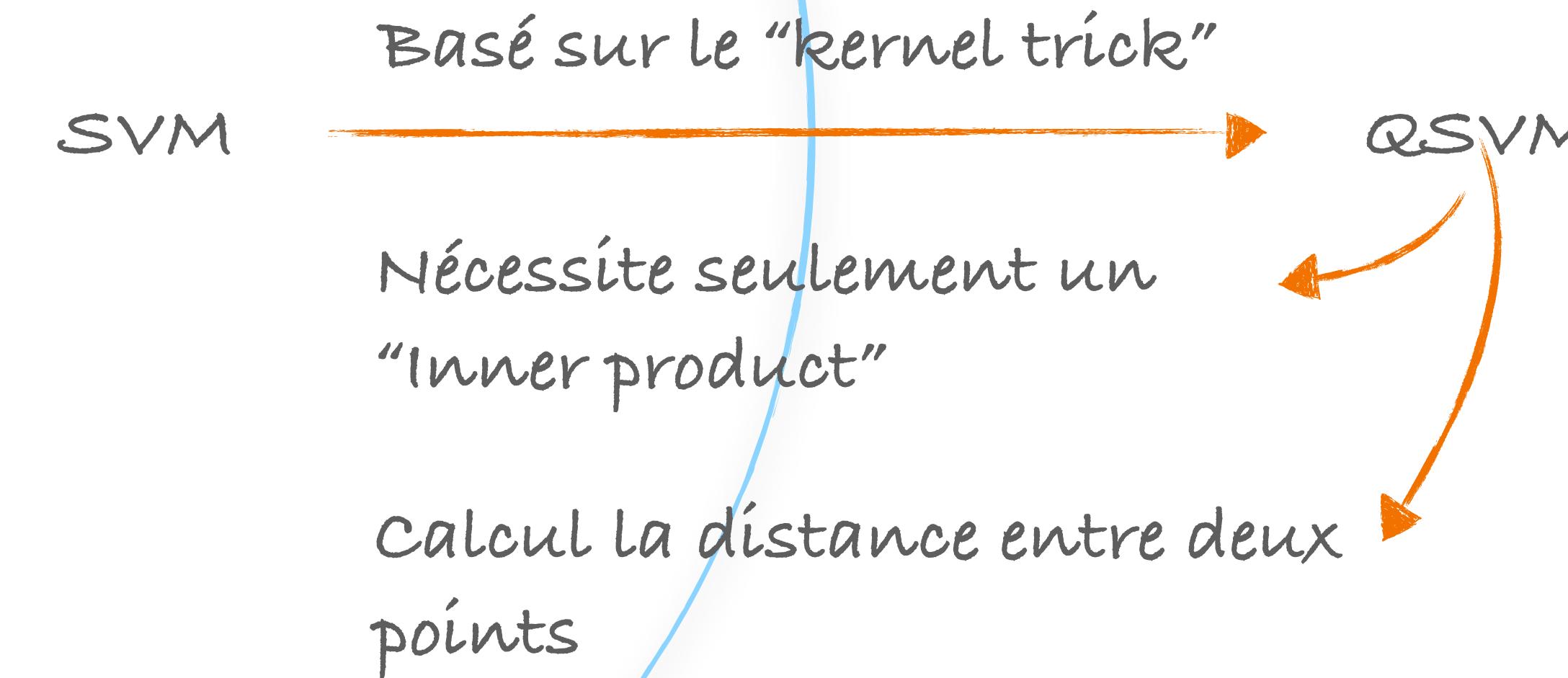
Mesure de similarité ou distance Euclidienne



Quantum Support Vector Machines

Quantum Support Vector Machines

Quantum?



Quantum Support Vector Machines

Quantum?

Étapes

1)

Prendre 2 vecteurs dans l'espace de données

2)

Projection dans L'espace de feature pour faire des états quantiques

3)

Calcul le inner product

SVM

Basé sur le "kernel trick"

Nécessite seulement un "inner product"

Calcul la distance entre deux points

QSVM

Circuit paramétrisé par les données à encoder

Quantum Support Vector Machines

Formulation

$\varphi(\vec{x}) = \Phi(\vec{x}) | 0 \rangle$

Feature Map

Circuit paramétrisé
dépendant des données

Quantum Support Vector Machines

Formulation

Feature Map

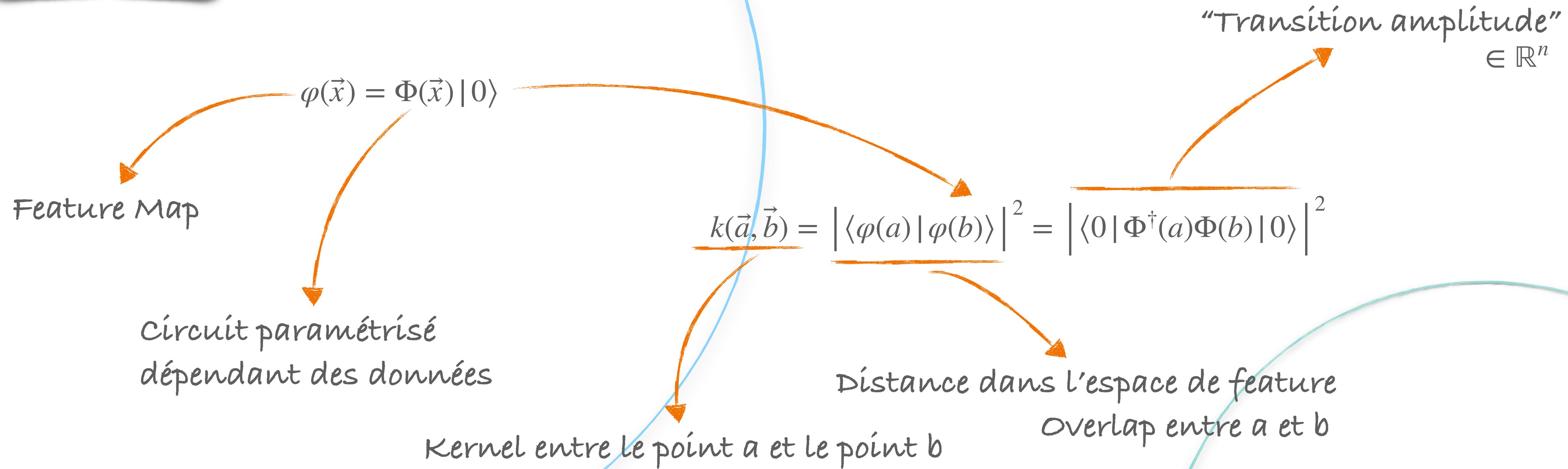
$$\varphi(\vec{x}) = \Phi(\vec{x}) | 0 \rangle$$

Circuit paramétrisé dépendant des données

$$k(\vec{a}, \vec{b}) = |\langle \varphi(a) | \varphi(b) \rangle|^2 = \left| \langle 0 | \Phi^\dagger(a) \Phi(b) | 0 \rangle \right|^2$$

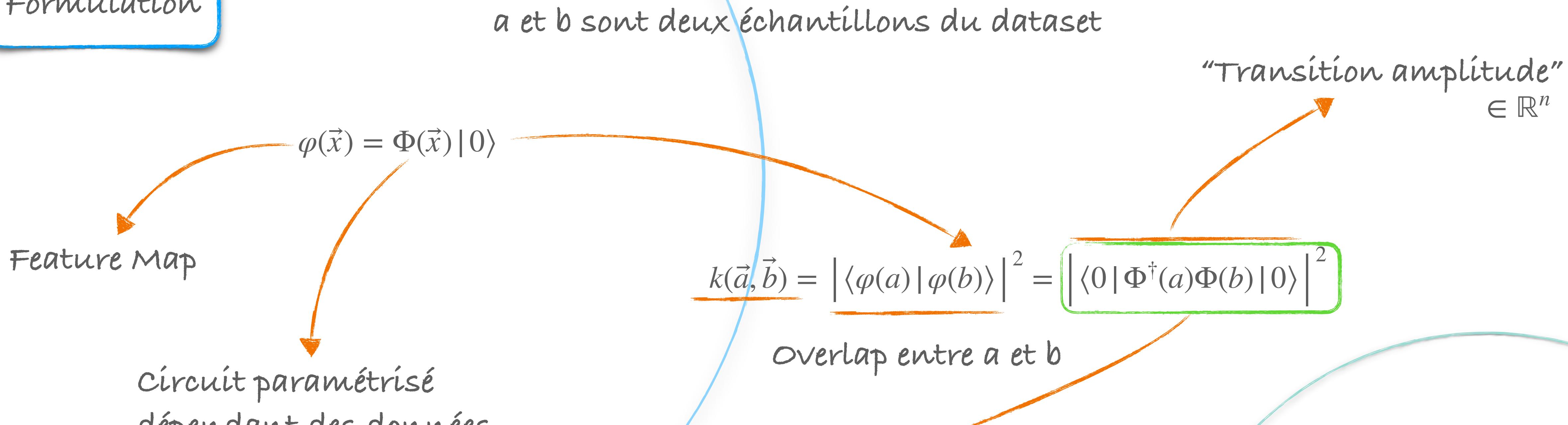
Quantum Support Vector Machines

Formulation



Quantum Support Vector Machines

Formulation



Quantum Support Vector Machines

Formulation

a et b sont deux échantillons du dataset

$$\varphi(\vec{x}) = \Phi(\vec{x}) |0\rangle$$

Feature Map

Circuit paramétrisé
dépendant des données

$$k(\vec{a}, \vec{b}) = |\langle \varphi(a) | \varphi(b) \rangle|^2 = |\langle 0 | \Phi^\dagger(a) \Phi(b) | 0 \rangle|^2$$

Overlap entre a et b

$$\text{Trivial sur un QC} \\ \Phi^\dagger(a) \Phi(b) |0\rangle$$

Mesure de la probabilité d'obtenir le state
 $|0\rangle$ avec le state $\Phi^\dagger(a) \Phi(b)$

$\Phi^\dagger(x)$ est l'inverse de $\Phi(x)$ comme on utilise des portes unitaires l'inverse correspond à écrire le circuit en partant de la droite vers la gauche

Kernels

Classical Kernel

A kernel function lives the feature space

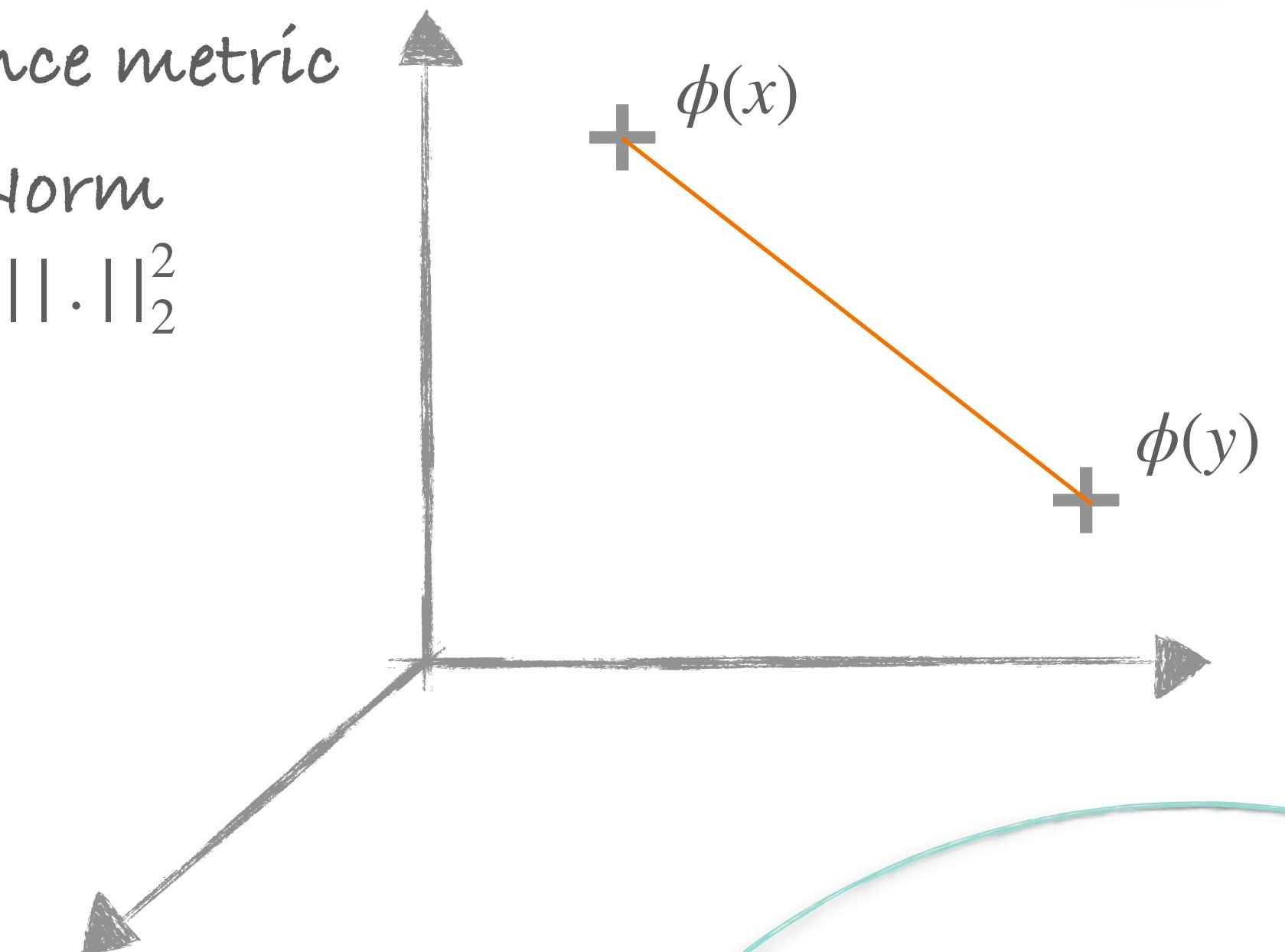
$$K : \Omega \times \Omega \rightarrow \mathbb{R}$$

$$K : (x, y) \rightarrow \langle \phi(x), \phi(y) \rangle$$

Euclidean distance/product

Distance metric

$$\text{Norm} \quad \| \cdot \|_2^2$$



Quantum Kernel

Compute the inner product (Hilbert-Schmidt) between matrices

$$\langle A, B \rangle_{HS} = \sum_{ij} A_{ij}^\dagger B_{ij}$$

$$|\phi(x)\rangle = U(x)|0\rangle$$

$$\Phi(x) = |\phi(x)\rangle\langle\phi(x)| = U(x)|0\rangle\langle 0|U^\dagger(x)$$

Projector / density matrix

$$K(x, y) = \langle \Phi(x), \Phi(y) \rangle_{HS}$$

$$= |\langle \Phi(x) | \Phi(y) \rangle|^2 \in \mathbb{R}^+$$

$$= |\langle 0 | U^\dagger(x)U(y) | 0 \rangle|^2 \in \mathbb{R}^+$$

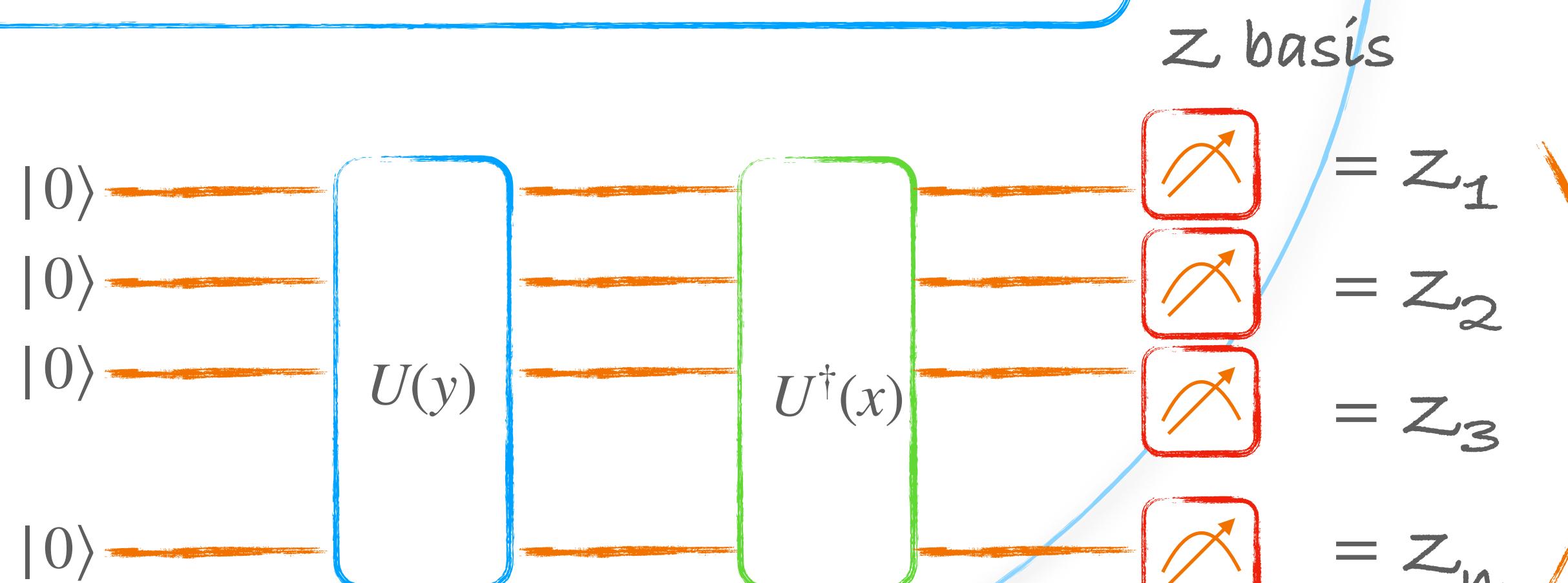
Transition amplitude

Quantum Support Vector Machines

Formulation

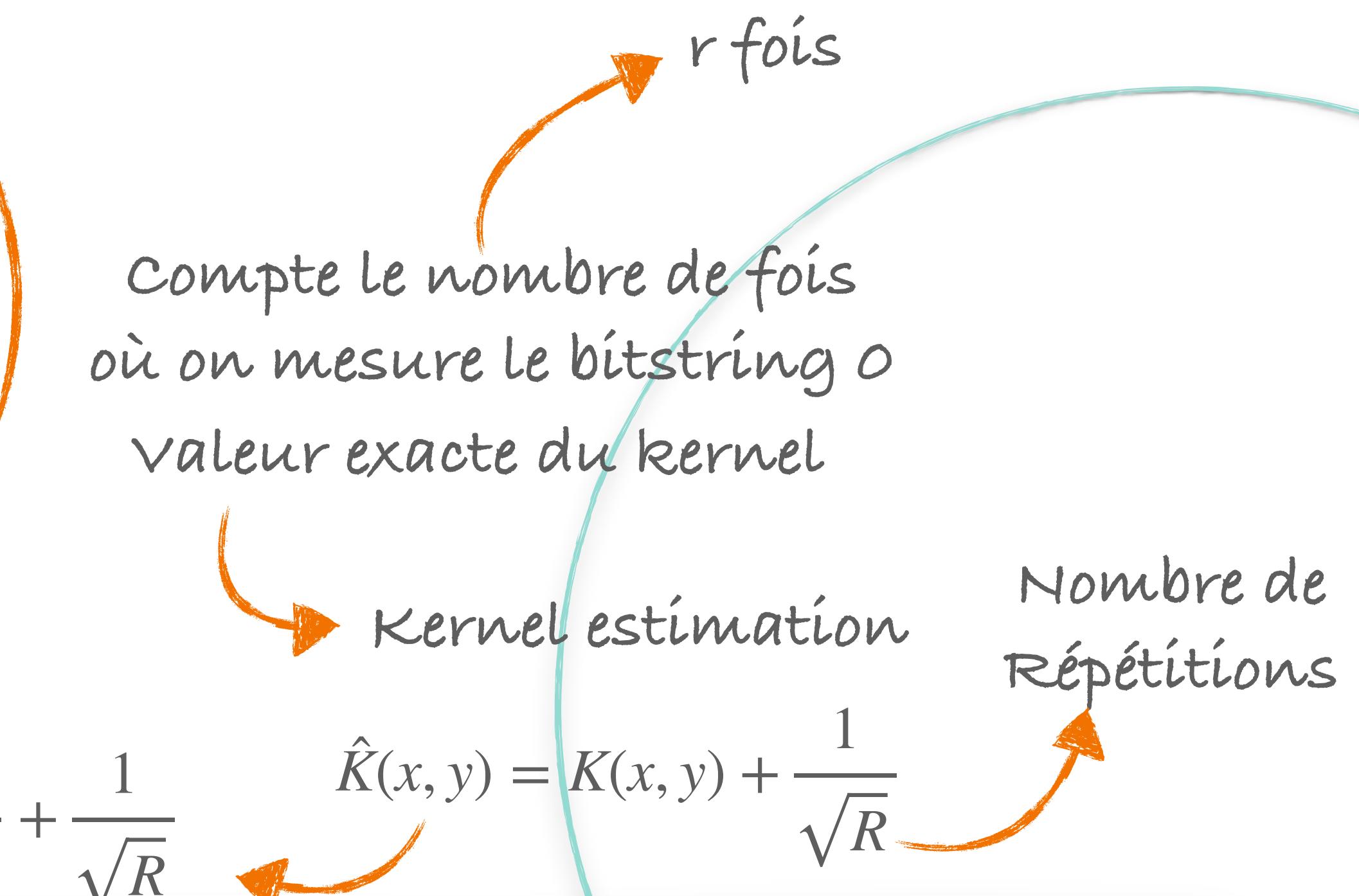
$$k(\vec{a}, \vec{b}) = |\langle \varphi(a) | \varphi(b) \rangle|^2 = |\langle 0 | \Phi^\dagger(a) \Phi(b) | 0 \rangle|^2$$

Mesure de la probabilité d'obtenir le state $|0\rangle$ avec le state $\Phi^\dagger(a)\Phi(b)$



Kernel estimation

Pour m échantillons on doit faire l'estimation m^2 fois



Quantum Support Vector Machines

code!!

Kernel avec les amplitudes

```
def kernel_circ(a,b):
    # phi a
    qml.AmplitudeEmbedding(
        a, wires=range(nqubits), pad_with=0., normalize = True
    )
    # phi dagger b
    qml.adjoint(qml.AmplitudeEmbedding(
        b, wires=range(nqubits), pad_with=0., normalize = True
    ))
    return qml.probs(wires=range(nqubits))
```

Quantum Support Vector Machines

code!!

Kernel avec les angles

```
def kernel_circ(a,b):
    # phi a
    qml.AngleEmbedding(
        a, wires=range(nqubits)
    )
    # phi dagger b
    qml.adjoint(qml.AngleEmbedding(
        b, wires=range(nqubits)
    ))
    return qml.probs(wires=range(nqubits))
```

Quantum Support Vector Machines

code!!

Notebook !

Références

- [1] Combarro E. F. & González-Castillo S., 2023, A Practical Guide to Quantum Machine Learning and Quantum Optimization, Chapitre 9
- [2] Schuld M. & Petruccione F., 2021, Machine Learning with Quantum Computers, Chapter 6