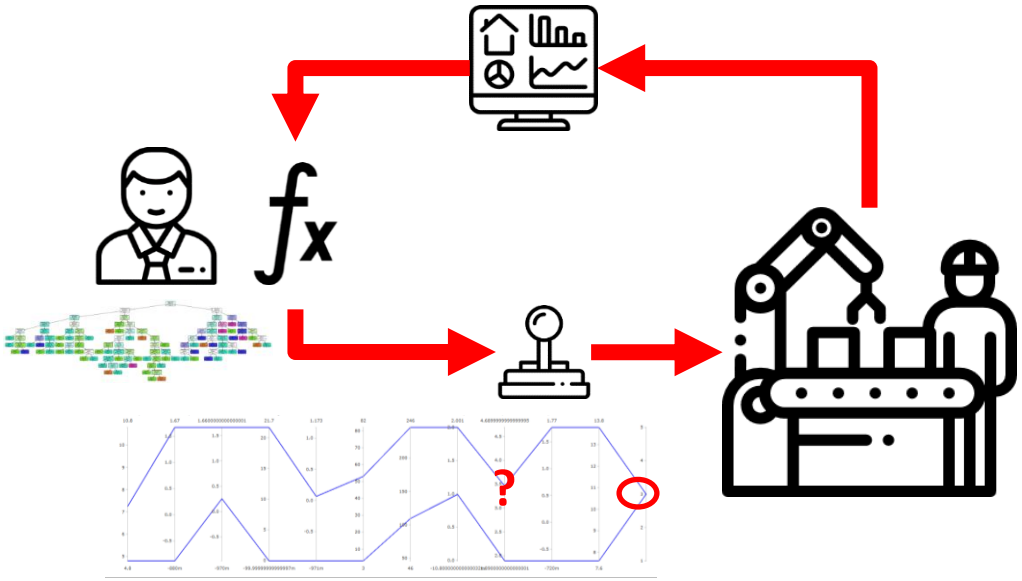
# Comment améliorer votre production via les techniques de Machine Learning?

# IA pour l'amélioration des performances des systèmes industriels




**Wahb ZOUHRI**
Jean-Yves DANTAN
Lazhar HOMRI
Alain ETIENNE

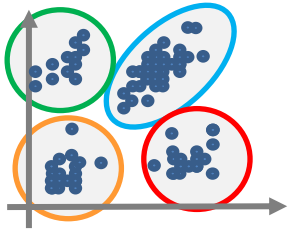
# Introduction

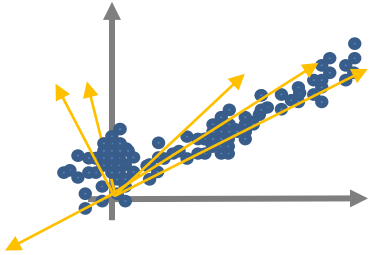
***Clustering***
Regrouper un ensemble d'objets en fonction de leurs caractéristiques.

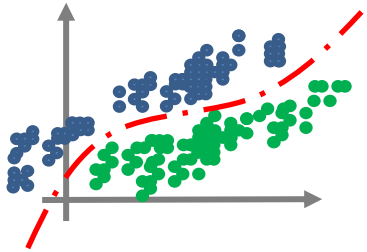
**Association**
Découvrir des relations entre variables.

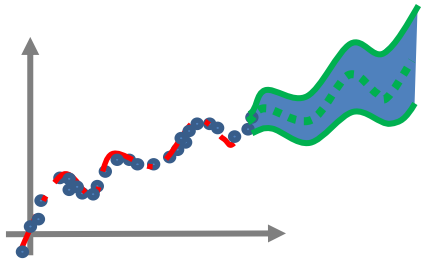
**Classification**
Prédire la catégorie (classe) d'une nouvelle observation.

**Regression**
Trouver une fonction qui modélise les données.
Prédire









**K-moyennes** ↻

**ACP** ☑

**Arbre de décision** ☑

**Logistic regression** ☑

**KNN** ↻

**SVM** ↻

**Réseaux de neurones** ?

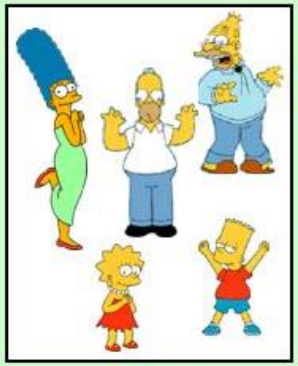
# 1. K-means

# Segmentation par K-moyennes (K-means)

## Introduction

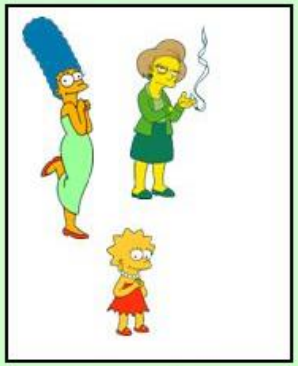
**Quels sont les regroupements possibles entre ces objets ?**

**Clustering**



Simpson's Family

School Employees

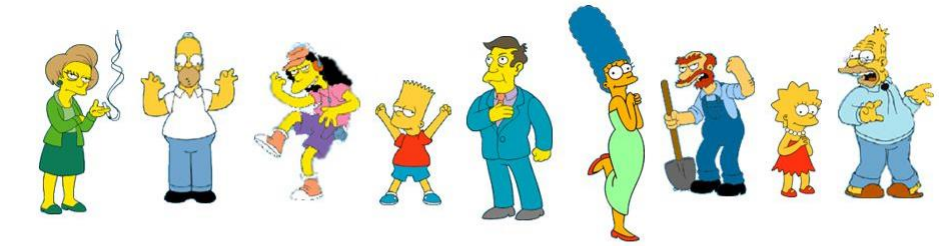


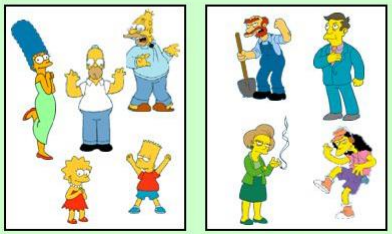
Females

Males

# Segmentation par K-moyennes (K-means)

## Introduction

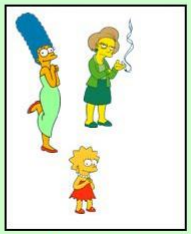




Simpson's Family



School Employees



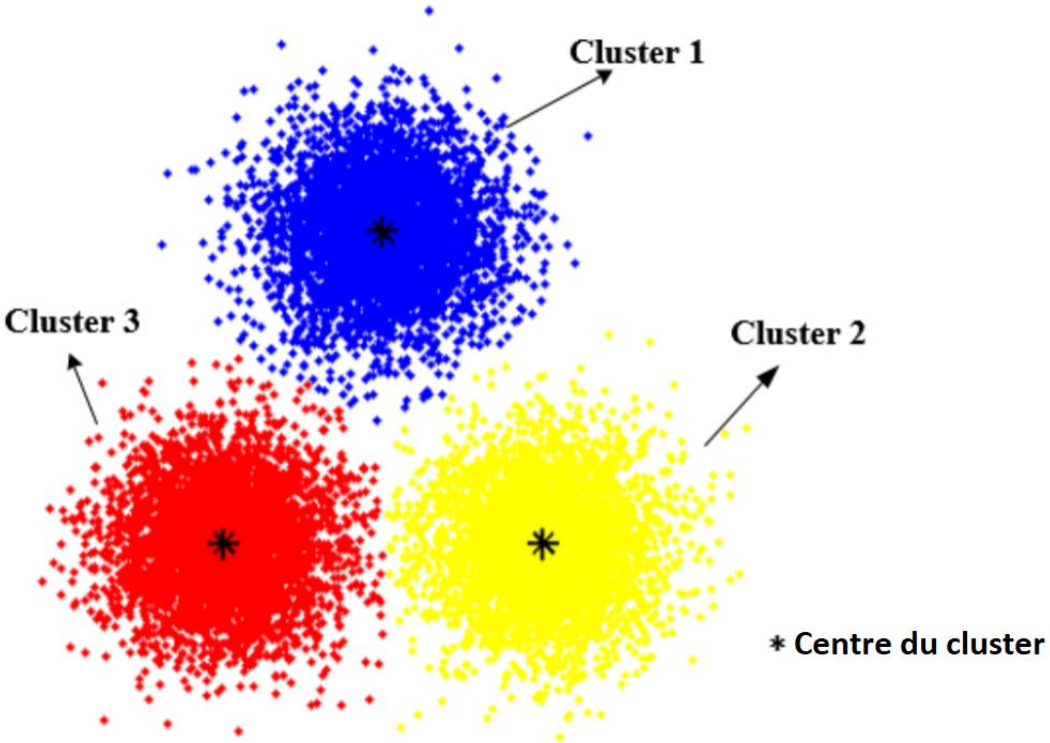
Females

Males

1. Qu'est-ce que le clustering ?
2. Comment déterminer les clusters ?
3. Comment pouvez-vous le faire efficacement ?

## Qu'est-ce que le clustering ?

Le **clustering** (la classification non supervisé) est la recherche d'une partition, ou répartition des individus en classes, ou catégories; Ceci est fait en optimisant un critère visant à regrouper les individus dans des classes, chacune le plus homogène possible et, entre elles, les plus distinctes possible, **c'est le principe de clustering.**



## Les Clusters

- A la base, un cluster est un ensemble d'éléments. Cet ensemble est différent des autres.
- Les méthodes d'analyse de clusters sont des **algorithmes non-supervisés**, ils permettent de générer et de trouver des classes naturelles.
- Les deux propriétés importantes définissant un cluster pertinent sont :
  1. **sa cohésion interne** (que les objets appartenant à ce cluster soient les plus **similaires** possibles).
  2. **son isolation externe** (que les objets appartenant aux autres clusters soient les plus éloignés possible).

**mais, qu'est-ce que la similarité ?**

## Concept de la similarité

Difficile à définir ! Mais nous la reconnaissons quand nous la voyons.

La véritable notion de similarité est une question philosophique. Nous adopterons une approche plus pragmatique.

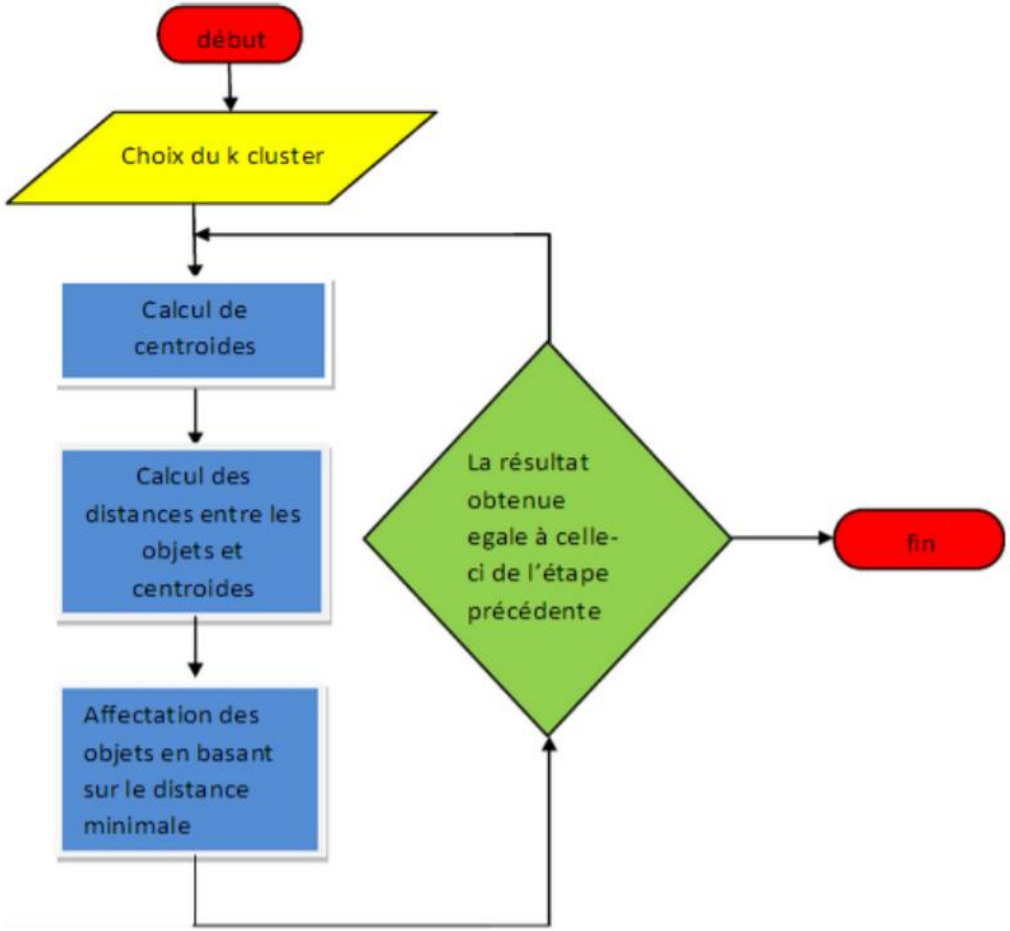
Pour de nombreux algorithmes/méthodes, il est plus facile de penser en termes de distance (plutôt que de similarité) entre les vecteurs. C'est bien le cas pour la méthode des **K-moyennes**.

# Comment déterminer les clusters ?

## K-means

**Principe :** L'algorithme de clustering K-means consiste à classer les objets en **K clusters**, ces clusters sont présentées par les moyennes pondérées des objets inclus dans les clusters, ces moyennes sont appelées ''**centroïdes**''.

**Stabilité de l'algorithme:** L'état stable est l'état dont lorsque l'algorithme trouve les mêmes résultats que celle-ci du l'étape précédente; C'est-à-dire, il n'y a plus de changement au niveau des clusters.



# Comment déterminer les clusters ?

## Étapes de la K-means

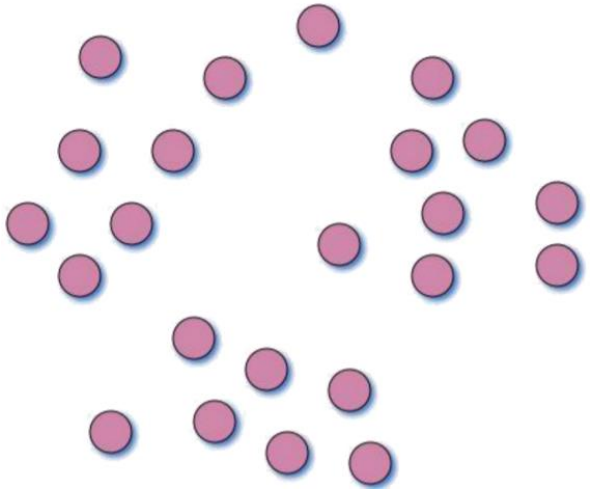
La méthode des k-moyennes est une méthode d'apprentissage non-supervisée : apprentissage à partir de données non étiquetées; c-à-d la sortie dénotée généralement par ''y'' n'est pas donnée.

Nous sommes donc face à un jeu de données défini comme une matrice de **n** observations (lignes) et **p** paramètres (colonnes) $X_1, X_2, \dots, X_p$.

| $X_1$ | $X_2$ | $X_3$ | $X_p$ | Y |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

sans étiquette

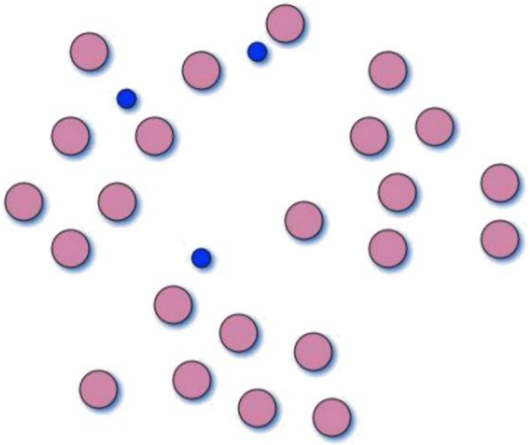
**Étape 1 :** Le choix d'un nombre **K** qui va présenter le nombre des classes (dans notre cas le **K** égale à 3).



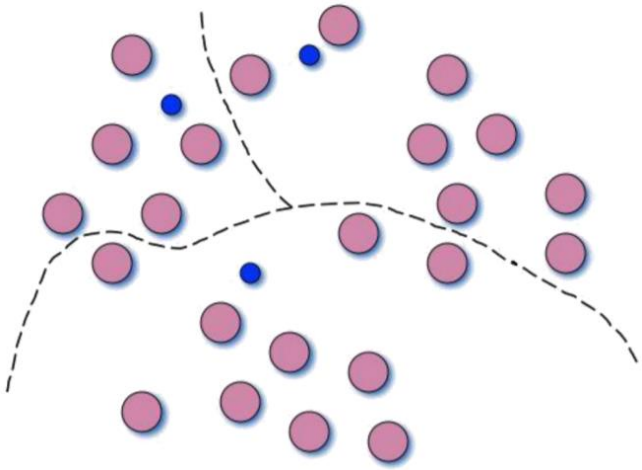
# Comment déterminer les clusters ?

## Étapes de la K-means

**Étape 2 :** Choix aléatoire de k points (centroïdes).



**Étape 3 :** Calculer les distances *(distance euclidienne)* entre les différents points et les centroïdes et affecter chaque objet au cluster qui convient en basant sur le principe de la distance minimale.



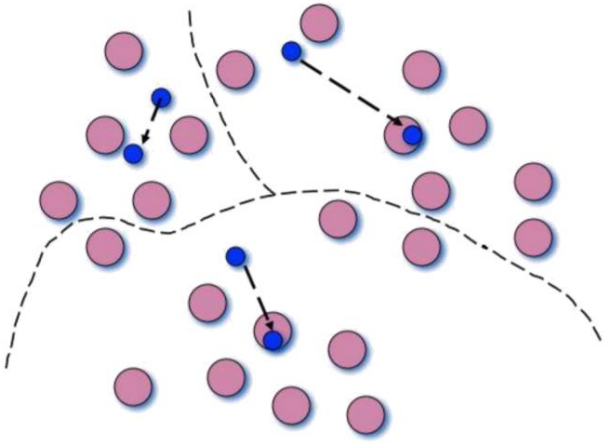
Distance euclidienne:

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

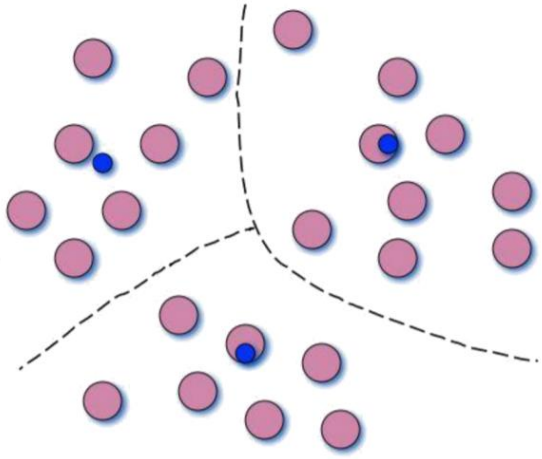
# Comment déterminer les clusters ?

## Étapes de la K-means

**Étape 4 :** Recalculer les valeurs des centroïdes comme étant les valeurs moyennes calculées pour chaque cluster.



**Étape 5 :** Répéter les étapes 3 et 4 jusqu'à ce qu'on trouve l'état stable..



# Comment pouvez-vous le faire efficacement ?

## La méthode du coude

**Choisir un nombre de cluster K n'est pas forcément intuitif**. Spécialement quand le jeu de données est grand et qu'on n'ait pas des hypothèses sur les données.

- Un valeur grande de **K** peut conduire à un partitionnement trop fragmenté des données. Ce qui empêchera de découvrir des patterns intéressants dans les données.
- Un nombre de clusters trop petit, conduira à avoir, potentiellement, des cluster trop généralistes contenant beaucoup de données. Dans ce cas, on n'aura pas de patterns "fins" à découvrir.

**Comment faire donc pour choisir un nombre de cluster qui permettra de mettre en lumière des patterns intéressants entre les données ?**

Malheureusement, il n'existe pas de processus automatisé pour trouver le nombre optimal de clusters, mais nous disposons de méthodes qui nous permettent de choisir un bon nombre de clusters, notamment la **méthode du coude**.

# Comment pouvez-vous le faire efficacement ?

## La méthode du coude

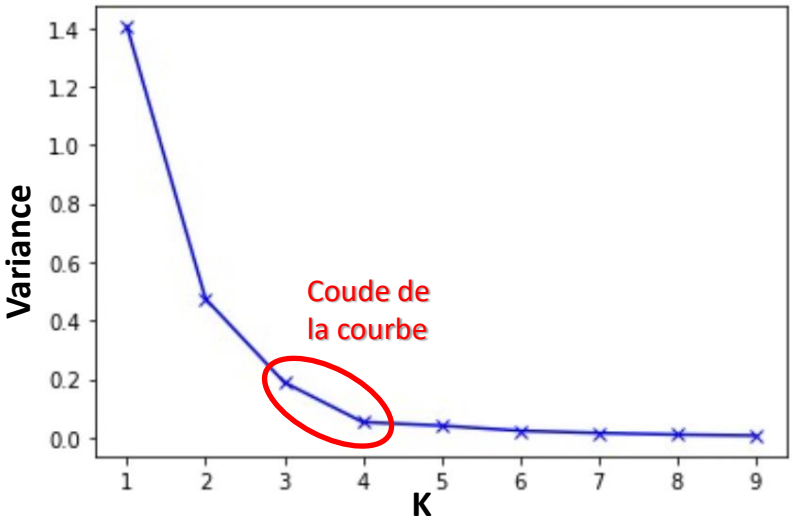
La méthode la plus usuelle pour choisir le nombre de clusters est de lancer **K-Moyennes avec différentes valeurs** de et de calculer la variance des différents clusters.

La variance est la somme des distances entre chaque *centroid* d'un cluster et les différentes observations inclues dans le même cluster:

$$V = \sum_j \sum_{x_i \to c_j} D(c_j, x_i)^2$$

Avec :

- $c_j$ : Le centre du cluster (le centroïd)
- $x_i$ : la ième observation dans le cluster ayant pour centroïd $c_j$
- $D(c_j, x_i)$ : La distance euclidienne entre le centre du cluster et le point $x_i$




le point du coude est celui du nombre de clusters à partir duquel la variance ne se réduit plus significativement.

## Avantages et inconvénients de K-means

Avantages :

- L'algorithme de k-means est très populaire du fait qu'il est très facile à comprendre et à mettre en œuvre.
- Sa simplicité conceptuelle et sa rapidité.

Inconvénients :

- N'est pas applicable en présence d'attributs où la moyenne n'est pas définie.
- Les clusters sont construits par rapports à des objets inexistants (les milieux).
- Les points aberrants sont mal gérés.

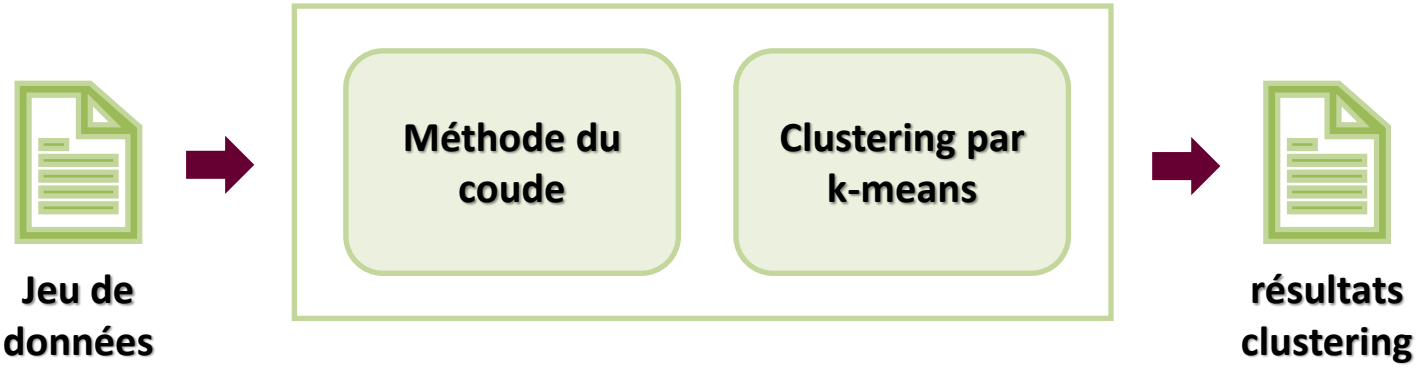
# Segmentation par K-moyennes (K-means)

## Application

L'**objectif** de cette 1ère application est d'appliquer la méthode des **K-means** sur un jeu de données industriel à l'aide des bibliothèques **Python**.

Cette application nécessite :

1. L'importation de jeu de données,
2. L'application de la méthode du coude pour identification d'un **K** optimal,
3. La création des clusters,
4. L'enregistrement des résultats sur un nouveau fichier Excel.

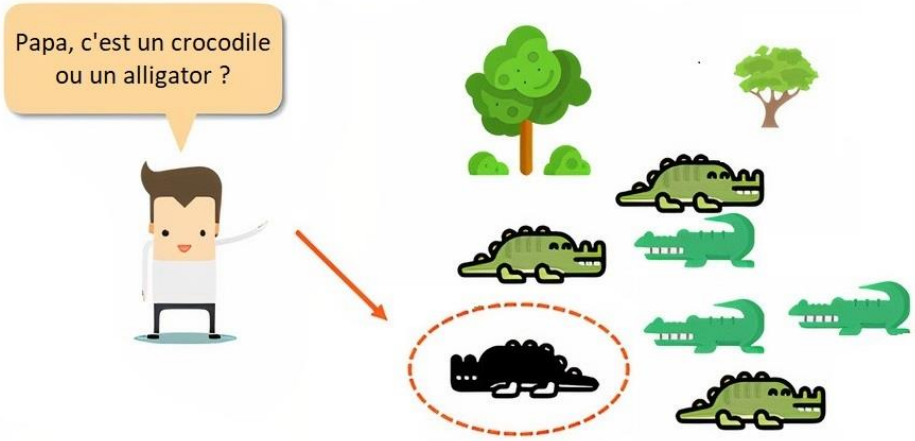


# 2. KNN

# Classification par KNN

## Introduction

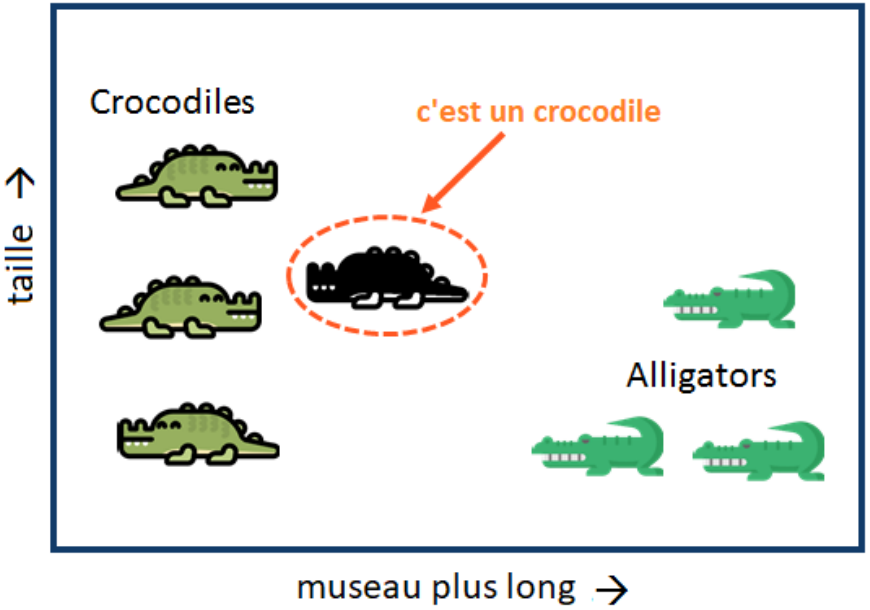





**Difference**

- les crocodiles sont plus grands que les alligators
- les crocodiles ont un museau plus petit

- les alligators sont de taille plus petite
- les alligators ont un museau plus long



# Classification par KNN

## Plan

**Dis-moi qui sont tes ~~amis~~ voisins et je te dirai qui tu es**

1. Qu'est-ce que la méthode des k plus proches voisins (**KNN**) ?
2. La différence entre **KNN** (classification) et **K-means** (clustering) ?
3. Comment cela fonctionne-t-il ?
4. Comment le faire efficacement ?

# Classification par KNN

## KNN, c'est quoi ?

- La
  d'a
  pro
- Il r
  trè
- Co
  obj
  ma

### Clustering

#### Concept de la similarité

Difficile à définir ! Mais nous la reconnaissons quand nous la voyons.

La véritable notion de similarité est une question philosophique. Nous adopterons une approche plus pragmatique.

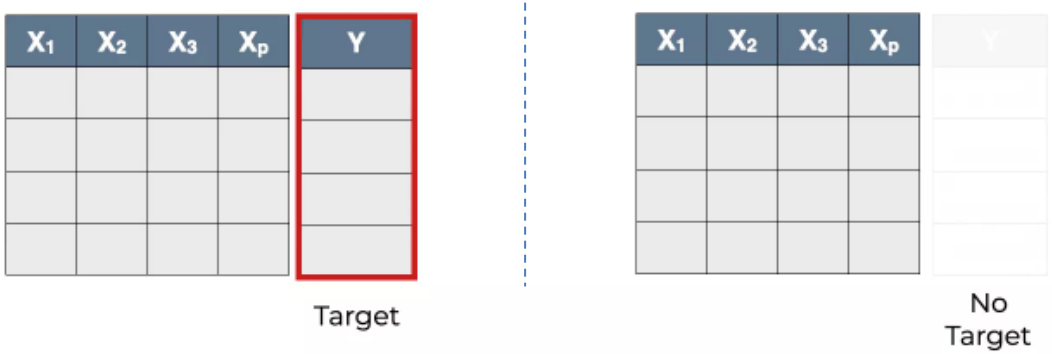
Pour de nombreux algorithmes/méthodes, il est plus facile de penser en termes de distance (plutôt que de similarité) entre les vecteurs. C'est bien le cas pour la méthode des **K-moyennes**.

**Mais, quelle(s) différence(s) y a-t-il entre les deux méthodes ?**

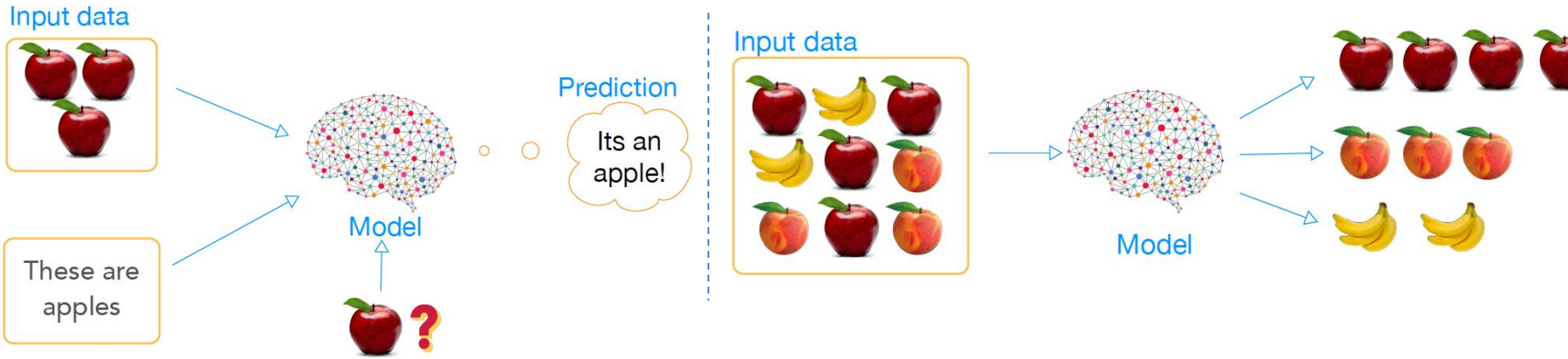
# Différence entre KNN et K-means

## KNN *vs* K-means

K-means est un algorithme **d'apprentissage non supervisé** utilisé pour les problèmes de **clustering**, tandis que KNN est un algorithme **d'apprentissage supervisé** utilisé pour les problèmes de **classification**.




La classification (ex: KNN) consiste essentiellement à mettre en place (à l'aide d'étiquettes) un modèle qui permet de prédire la classe d'une nouvelle donnée. D'un autre côté, le clustering consiste à regrouper des points de données similaires, sans avoir besoin de classes précises.



# Comment cela fonctionne-t-il ?

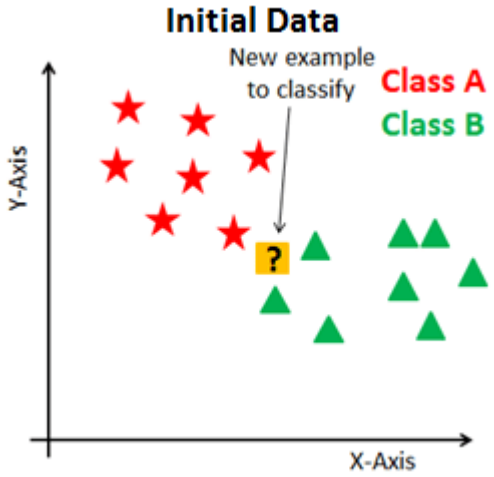
## Étapes de KNN

Au départ, nous sommes face à un jeu de données défini comme une matrice de **n** observations (lignes) et **p** paramètres (colonnes) $X_1, X_2, \dots, X_p$ auxquels on ajoute le paramètre **Y** qui représente les étiquettes.

| $X_1$ | $X_2$ | $X_3$ | $X_p$ | Y |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

étiquettes

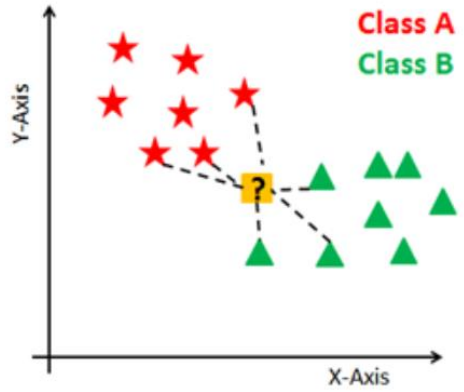
**Étape 0 - Objectif** : Prédire la classe d'une nouvelle donnée.



# Comment cela fonctionne-t-il ?

## Étapes de KNN

**Étape 1 :** Calculer la distance *(ex: distance euclidienne)* entre la nouvelle donnée à prédire et les différents points du jeu de données.

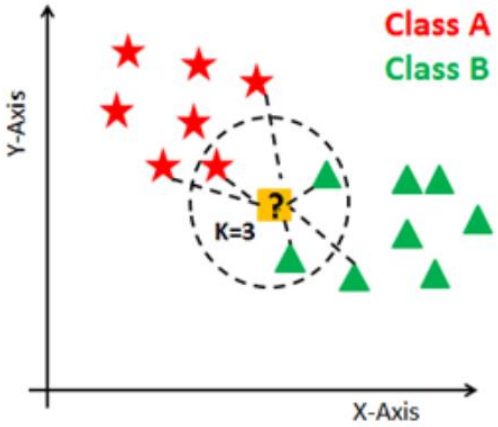



Distance euclidienne:

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

**Étape 2 :** Choisir le nombre **K** de voisins à considérer,

*(K=3 dans cet exemple).*

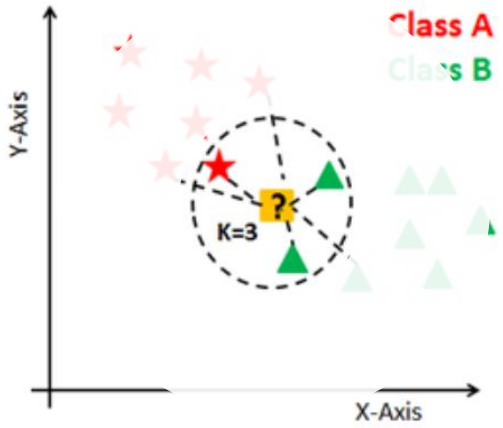


# Comment cela fonctionne-t-il ?

## Étapes de KNN

**Étape 3 :** Parmi ces K voisins :

- comptez le nombre de données de chaque classe.
- attribuer à la nouvelle donnée la classe majoritaire des voisins.




**Classe A = 1 voisin**
**Classe B = 2 voisins**

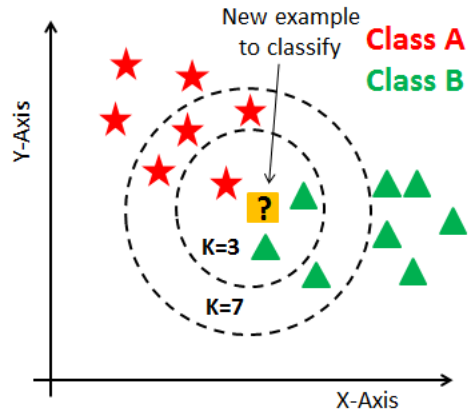
? =

Notre modèle **KNN** est prêt ☺ **…**

**… cependant, comment choisir la valeur optimale du paramètre K ?!!** ☹

# Comment pouvez-vous le faire efficacement ?

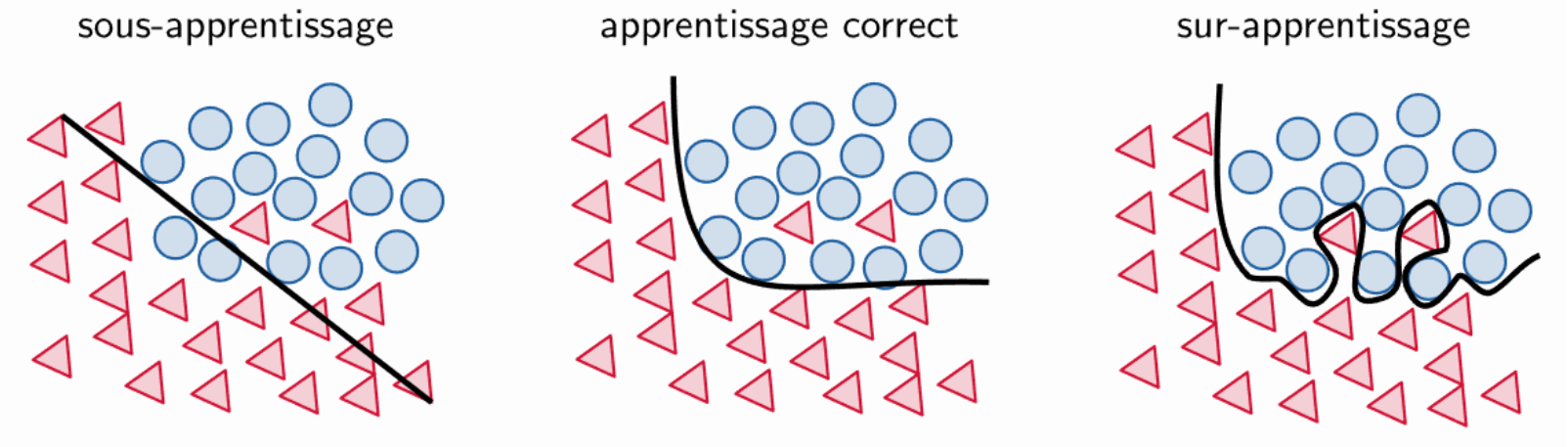
## Optimiser le seul hyperparamètre K




pour **K = 3** ? = ▲

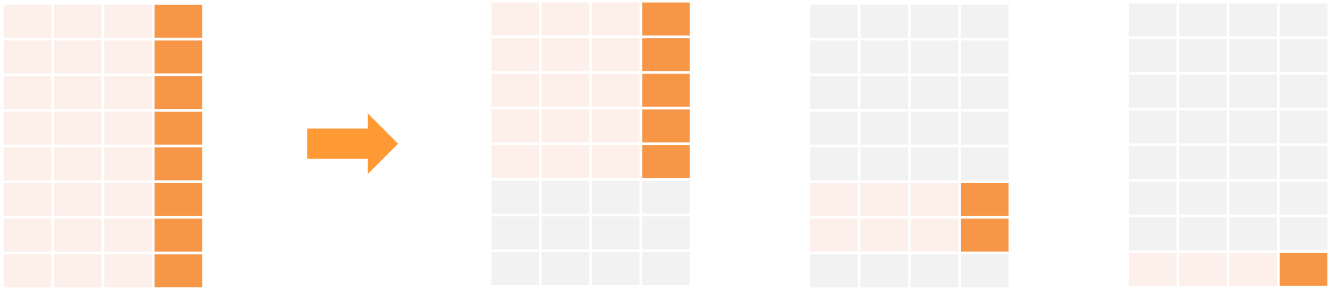
pour **K = 7** ? = ★

- Il n'existe pas de méthodes statistiques prédéfinies pour trouver la valeur optimale de **K**.
- Le choix d'une petite valeur de **K** conduit à des limites de décision instables.
- Une valeur **K** élevée est meilleure pour la classification car elle permet de lisser les limites de décision.



# Comment pouvez-vous le faire efficacement ?

## Optimiser le seul hyperparamètre K



Jeu de données

Apprentissage

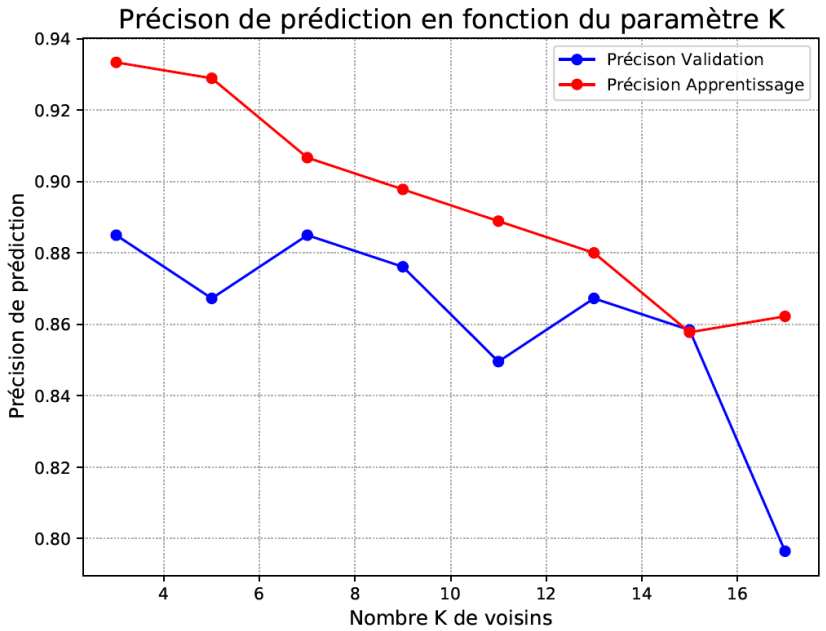
Validation

Test

| Créer le modèle | Définir les bons paramètres | Tester la généralisation du modèle |
|---|---|---|

## Dans le cas de KNN :

1. **Définir un intervalle de variation du paramètre K.**
2. **Pour chaque valeur de K :**
   - Entrainer un modèle KNN en utilisant le jeu d'apprentissage.
   - Calculer la précision de prédiction sur le jeu de validation.
3. **Choisir le K qui maximise la précision de prédiction.**



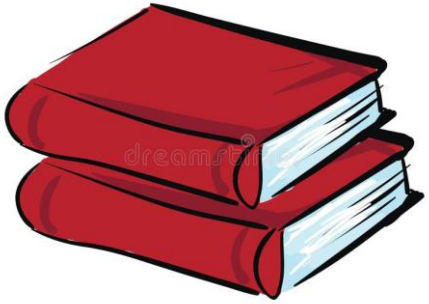
# Classification par KNN

## Informations complémentaires

- **KNN** est **non-paramétrique** car il ne fait aucune hypothèse sur les données étudiées.
- **KNN** est ce que nous appelons un algorithme **d'apprentissage paresseux** *(lazy learning)* : c'est un algorithme simple qui **mémorise** tous les cas disponibles et classe les nouvelles données ou les nouveaux cas sur la base d'une mesure de **similarité**.
- Les classificateurs paresseux sont plus utiles pour les grands jeux de données qui changent continuellement et qui sont couramment consultés.

## Applications réelles de KNN

1. De nombreuses entreprises font des recommandations personnalisées à leurs clients.
2. KNN peut rechercher des documents sémantiquement similaires en les considérant comme des vecteurs.
3. KNN peut être utilisé efficacement dans la détection de la fraude par carte de crédit…

## Avantages et inconvénients de KNN

Avantages :

- Algorithme simpliste et facile à utiliser
- Ne fait pas d'hypothèses sur les données.
- Méthode efficace pour les petits jeux de données.

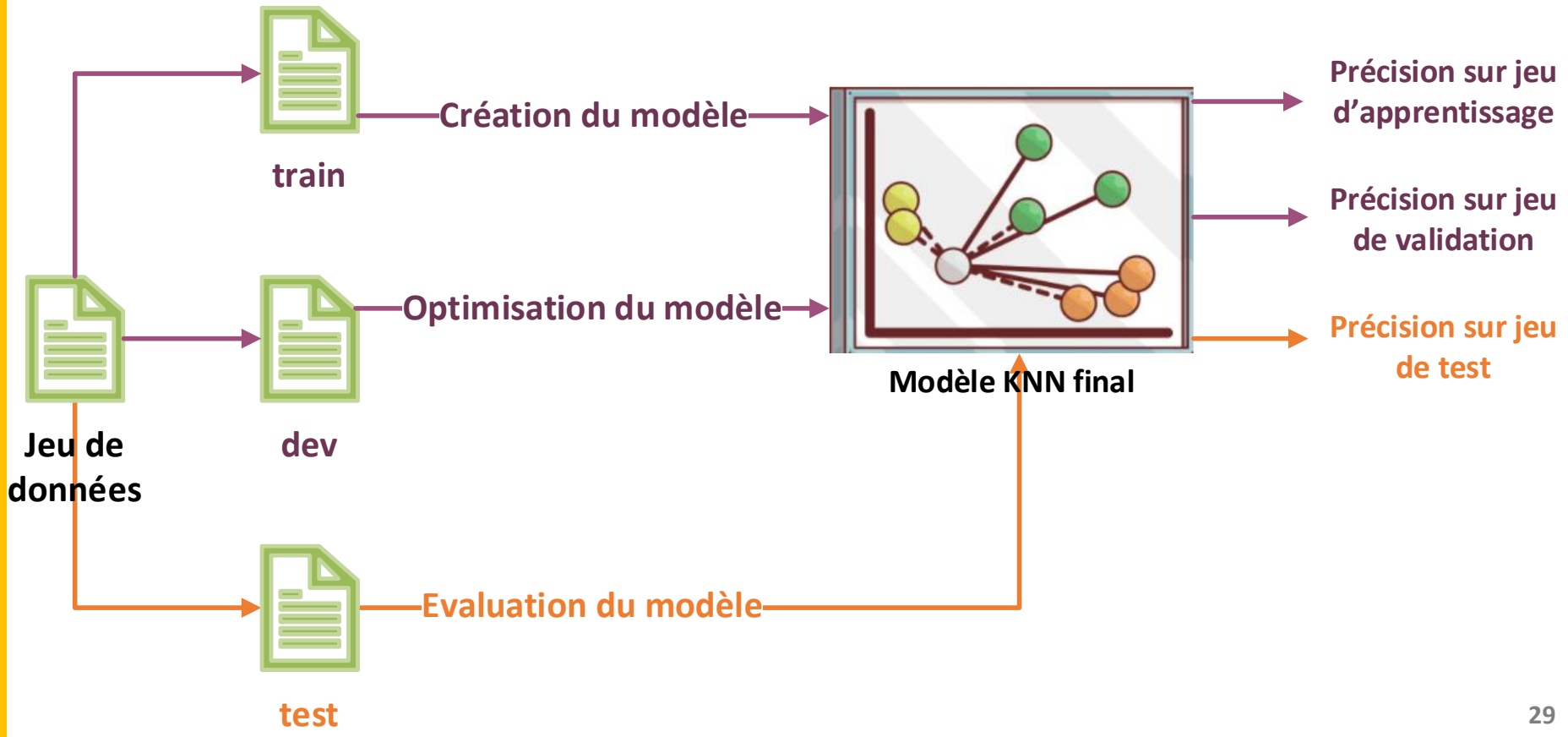
Inconvénients :

- Les grands jeux de données sont plus longs à traiter.
- Les données bruyantes peuvent entraîner une sur- ou sous-adaptation des données.

## Application

L'**objectif** de la 2ème application est de :

1. **Créer** un modèle prédictif **KNN** à partir du jeu d'apprentissage,
2. **Valider** les performances prédictives du modèle à l'aide du jeu de validation.
3. **Evaluer** la généralisation du modèle à l'aide du jeu de test.



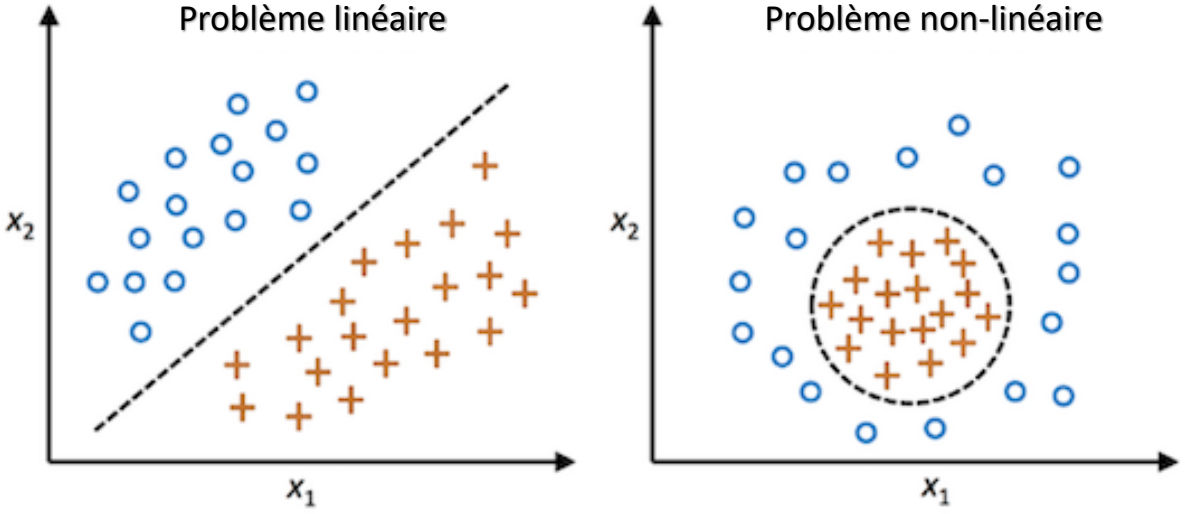
# 3. SVM

## Introduction

La meilleure façon de présenter les machines à vecteurs de support (SVM) est de considérer la simple tâche de classification binaire:

- le taux de change d'une devise évoluera à la **hausse** ou à la **baisse**, en fonction des données économiques.
- la décision d'**accorder** ou de **refuser** un prêt à un client sur la base d'informations financières personnelles.
- la prédiction de la **conformité** ou de la **non-conformité** d'un produit en fonction des paramètres opératoires.

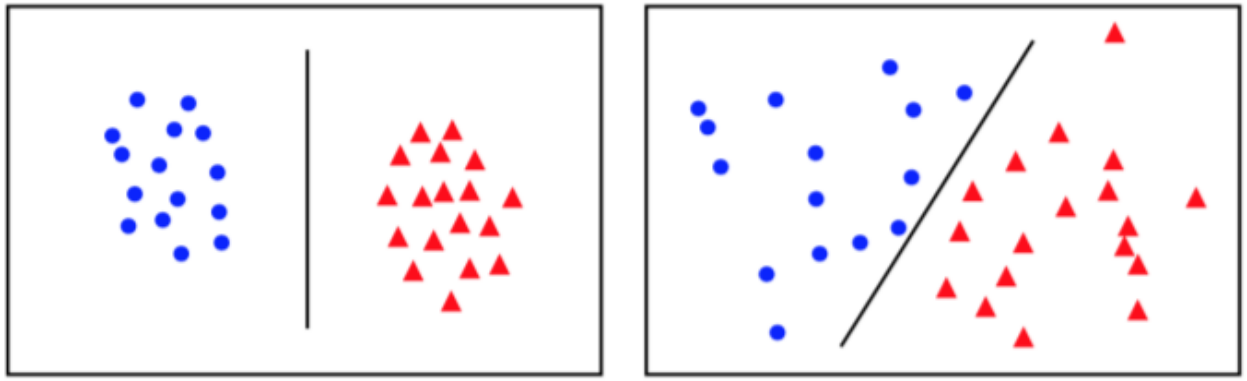
Étant un algorithme d'apprentissage supervisé, le principe de la SVM et d'apprendre à partir d'un jeu de données et de tenter de **généraliser** et de faire des prédictions correctes sur de nouvelles données.



## Cas linéaire séparable

Soit $\boldsymbol{B_a = \{\{x_n, y_n\} \in \mathbb{R}^d \text{ x } \{-1, 1\}\}_{n=1\ldots N}}$ un ensemble d'exemples d'apprentissage labellisés.

Les points $\{x_n, y_n\}$ sont linéairement séparables si il existe un **hyperplan** qui permet de discriminer correctement l'ensemble des données.



L'objectif de la SVM est d'apprendre une fonction de décision $\boldsymbol{f(x)}$, telle que:

$$f(x_n)\begin{cases} > 0 & si \quad y_n = 1 \\ < 0 & si \quad y_n = -1 \end{cases}$$

de manière équivalente $\boldsymbol{y_n f(x_n) > 0}$ pour tout exemple correctement prédit.

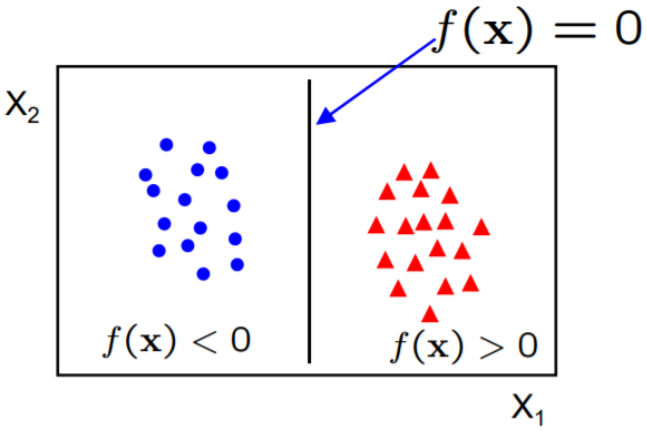
# SVM - Cas linéaire

## Fonction de décision

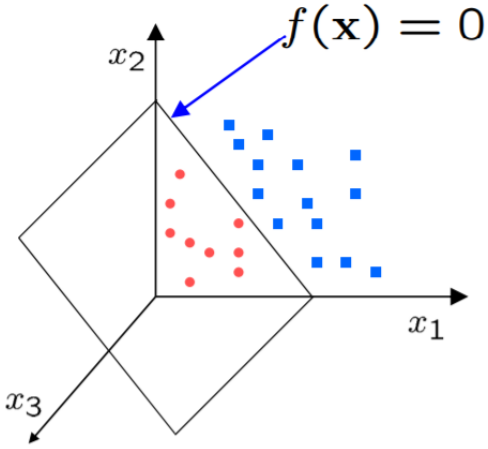
L'expression de la fonction de décision est :

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

- **w** est le vecteur normal à l'hyperplan, et **b** le biais.
- **w** est connu comme le vecteur de poids.




**Hyperplan en 2D = Droite**



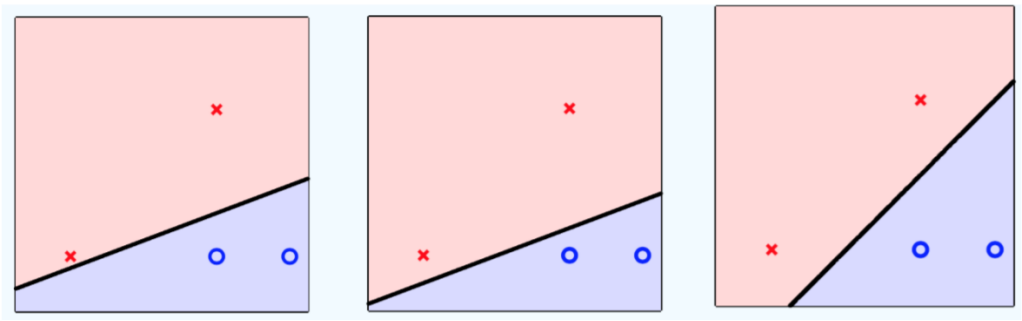

**Hyperplan en 3D = Plan**

L'hyperplan est l'ensemble de points $\boldsymbol{x_n}$ tel que : $\mathbf{w^T x_n + b = 0}$

# SVM – Cas linéaire

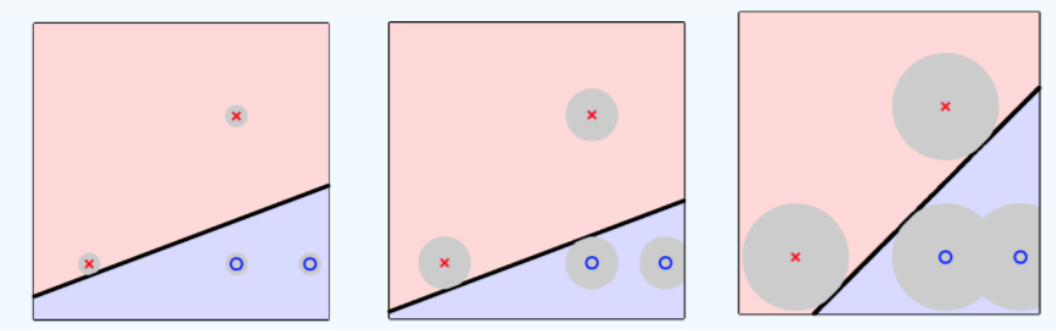
## Hyperplan séparateur

Il existe une infinité de lignes permettant de séparer les données qui sont linéairement séparables.

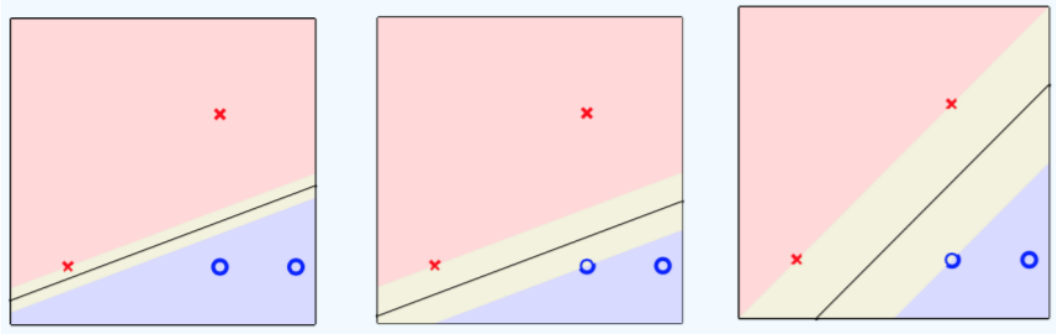


**Lequel choisir ? :**

La SVM choisit l'hyperplan qui tolère au maximum le bruit dans les données d'apprentissage.

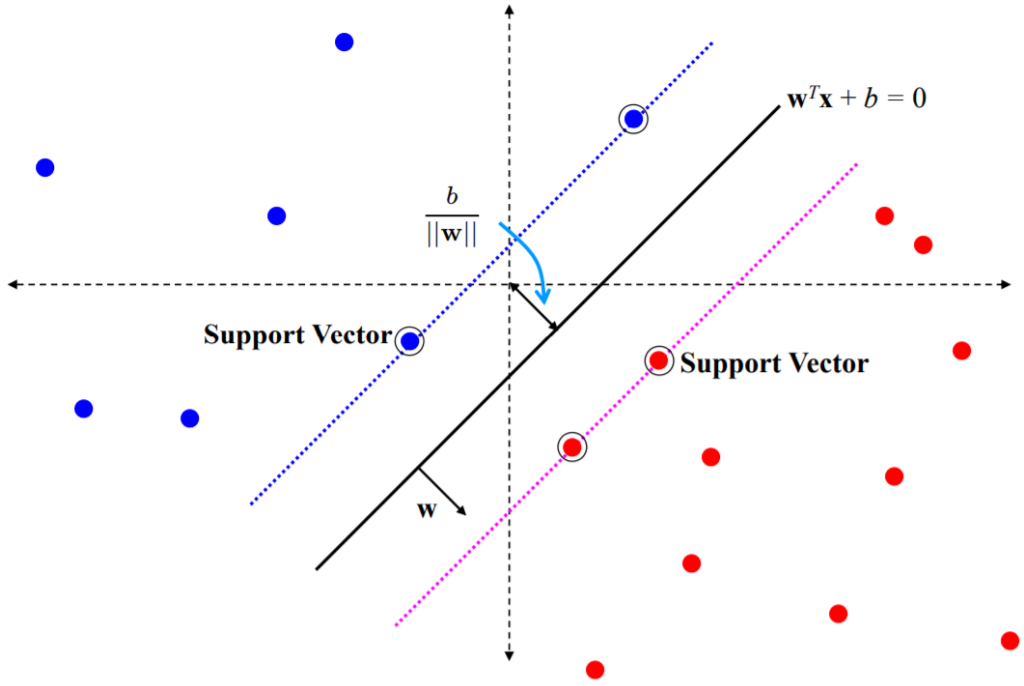


Ceci se traduit par une **marge maximale** entre les deux classes.



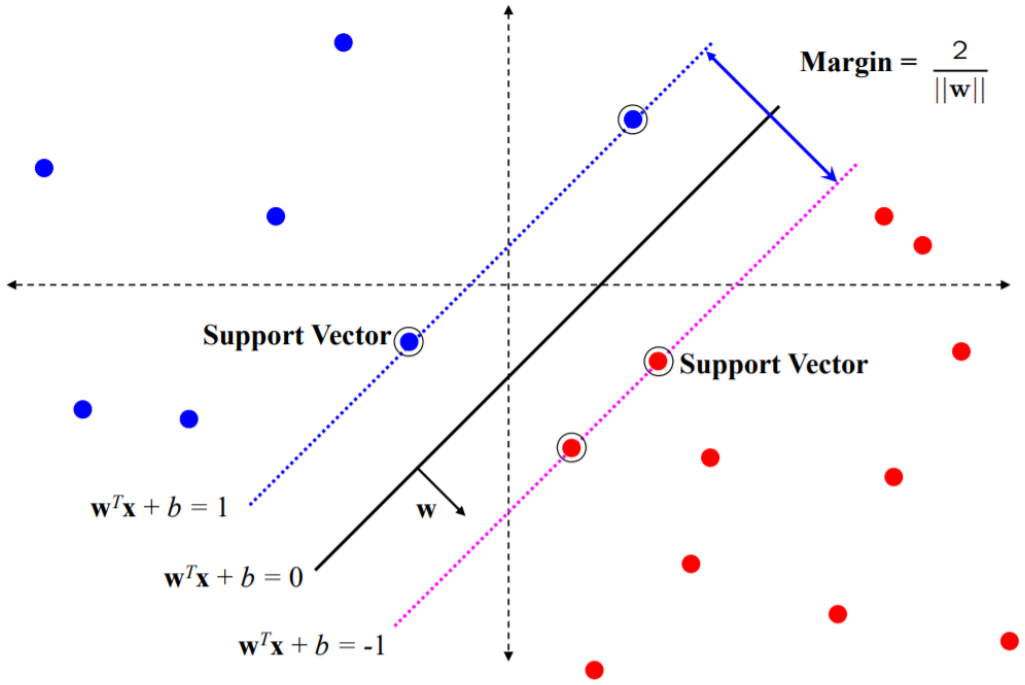
# SVM - Cas linéaire

## Récapitulatif des notions de la SVM




- La fonction de décision est définie par : $\mathbf{f(x_n) = w^T x_n + b}$
- L'hyperplan est l'ensemble de points $\{\mathbf{x_n}\}$ qui vérifient : $\mathbf{w^T x_n + b = 0}$
- Les **vecteurs de support** *(support vectors en anglais)* sont des points de données qui sont plus proches de l'hyperplan et qui influencent la position et l'orientation de l'hyperplan.

## Récapitulatif des notions de la SVM




- Les vecteurs de support définissent deux hyperplans $\{H^+, H^-\}$.
- Ces vecteurs de support vérifient les égalités suivantes :
  - Pour l'ensemble de $\mathbf{x}_+ \in \mathbf{H}^+ : \mathbf{w}^T\mathbf{x}_+ + \mathbf{b} = 1$
  - Pour l'ensemble de $\mathbf{x}_- \in \mathbf{H}^- : \mathbf{w}^T\mathbf{x}_- + \mathbf{b} = -1$
- La marge est donc définie comme la distance entre $H^+$ et $H^-$ selon le vecteur unitaire $\frac{\vec{w}}{\|w\|}$:

$$Marge = \frac{2}{\|w\|}$$

## Formulation de SVM linéaire

L'objectif de la SVM est de maximiser la marge; ceci peut être formuler comme un problème d'optimisation :

$$
\begin{aligned}
max \quad & \frac{1}{\|w\|} \\
s.t \quad & y_i(w^T.x_i + b) \geq 1 \qquad \forall\, i = 1, \dots, N \\
& y_i \in \{-1,1\} \\
& w \in \mathbb{R}^d; b \in \mathbb{R}
\end{aligned}
$$

La formulation quadratique équivalente à cette formulation :

$$
\begin{aligned}
min \quad & \frac{1}{2} w^T.w \\
s.t \quad & y_i(w^T.x_i + b) \geq 1 \qquad \forall\, i = 1, \dots, N \\
& y_i \in \{-1,1\} \\
& w \in \mathbb{R}^d; b \in \mathbb{R}
\end{aligned}
$$

Il s'agit d'un problème d'optimisation quadratique soumis à des contraintes linéaires, **il existe donc un minimum unique** (un seul vecteur w).

## Formulation de SVM linéaire

La formulation quadratique équivalente à cette formulation :

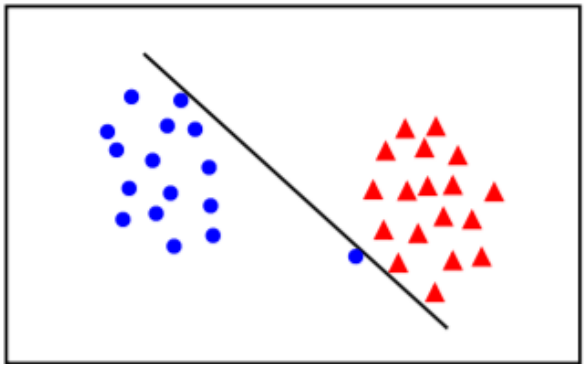
$$\begin{aligned} min \quad & \frac{1}{2} w^T . w \\ s.t \quad & y_i (w^T . x_i + b) \geq 1 \qquad \forall\, i = 1, \dots, N \\ & y_i \in \{-1, 1\} \\ & w \in \mathbb{R}^d ; b \in \mathbb{R} \end{aligned}$$

À l'aide des langages de programmation (e.g. python) nous pourrions résoudre ce problème d'optimisation et trouver les valeurs de **w** et **b**.

La prédiction d'une nouvelle donnée $\mathbf{x_n}$ se fait en évaluant le signe de la fonction de décision $\mathbf{f(x_n)}$:

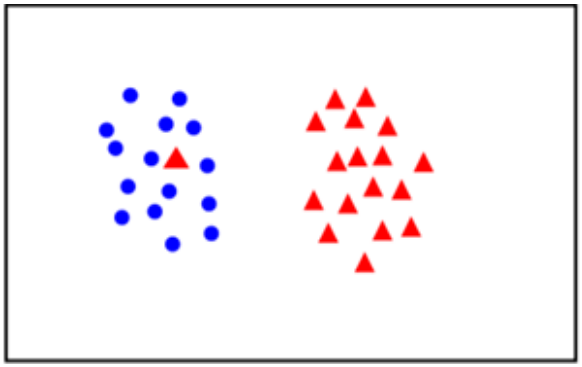
- si $\mathbf{f(x_n) = w^T x_n + b < 0}$ → classe négative $(y_n = -1)$
- si $\mathbf{f(x_n) = w^T x_n + b > 0}$ → classe positive $(y_n = +1)$

## Limitations de la SVM linéaire



Les points peuvent être séparés linéairement, mais la marge est très étroite.

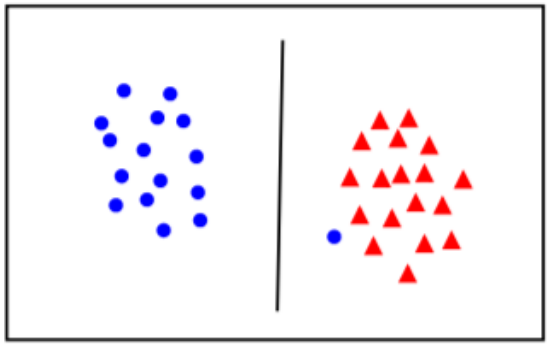
**Modèle très sensible au bruit !**

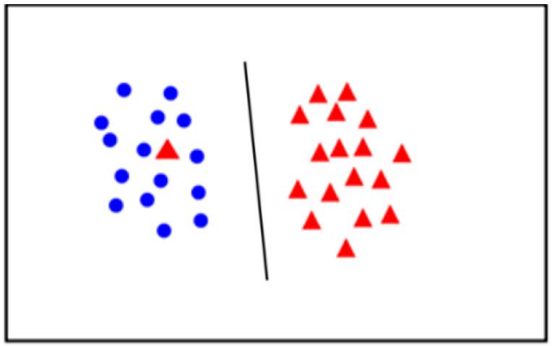


La SVM n'arrive pas à trouver un modèle qui sépare parfaitement les données.

**Algorithme ne converge pas !**

**Et si nous acceptons que certains exemples se trouvent du mauvais côté de l'hyperplan ?**





→ Meilleure généralisation du modèle sur les nouvelles données.

# SVM - Cas linéaire

## Marge poreuse

Pour certains jeux de données non linéairement séparables, un séparateur linéaire donnerait pourtant de bons résultats.

**Condition** : accepter que certains exemples ne respectent pas la contrainte.

**Problème** : la formulation actuelle des SVM ne le permet pas (*marge dure*).

**Solution** :

- Relâcher les contraintes
  - accepter que la marge soit franchie.
  - accepter que certains exemples soient du mauvais côté de l'hyperplan.
- Pénaliser ces relâchements dans la fonction à optimiser.

## Marge poreuse

Pour pénaliser les relâchements, une variable d'écart $\boldsymbol{\xi_n}$ (*slack variable*) est associée à chaque exemple d'apprentissage $(\boldsymbol{x_n}, \boldsymbol{y_n})$.

**Nouvelle contrainte :**

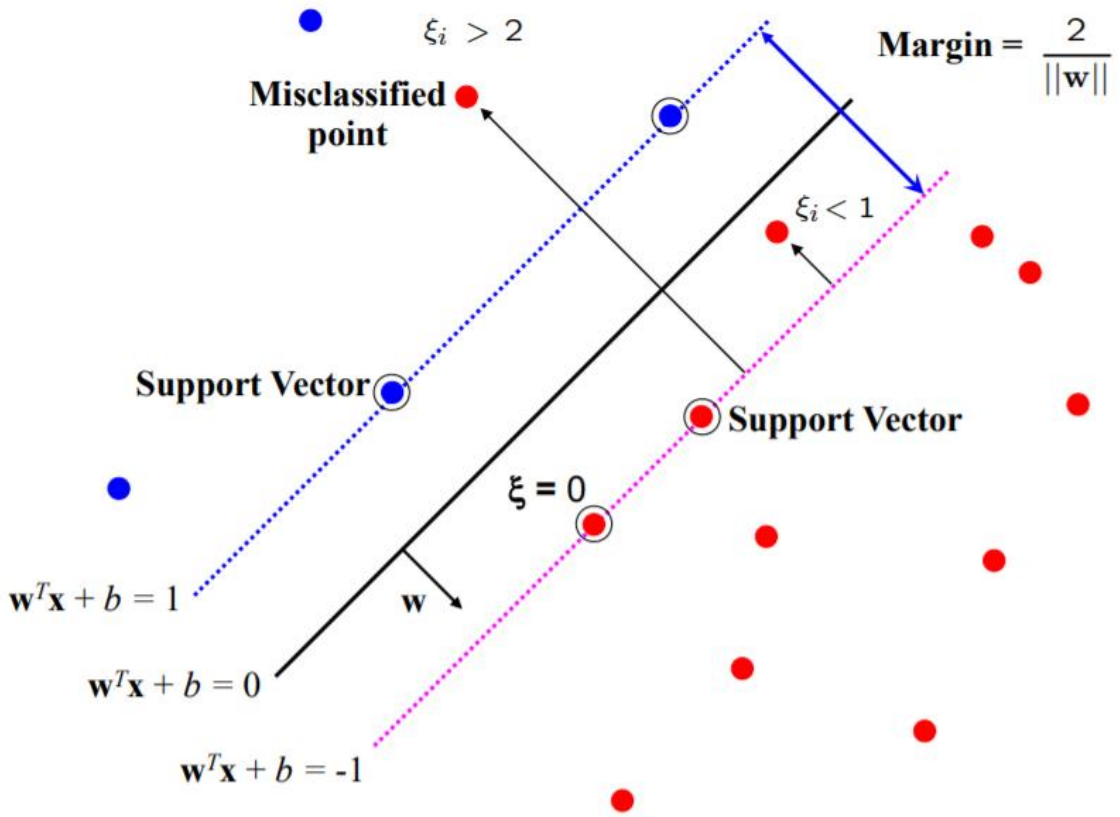
$$\boldsymbol{y_n}\left(\mathbf{w^T x_n} + \mathbf{b}\right) > \mathbf{1} - \boldsymbol{\xi_n}$$

$\xi_n$ renseigne à quel point un exemple $(\boldsymbol{x_n}, \boldsymbol{y_n})$ viole la contrainte :

- Si $\xi_n = 0$ l'exemple respecte la contrainte.
- Si $\xi_n > 1$ l'exemple est mal classé.
- Si $0 < \xi_n < 1$ l'exemple est bien classé mais il a franchi la marge.

# SVM - Cas linéaire

## Marge poreuse




**Nouvel objectif :**

On souhaite que $\sum \boldsymbol{\xi_n}$ soit minimal !

## Marge poreuse

La **nouvelle formulation** de SVM à marge poreuse est la suivante:

$$
\begin{aligned}
min \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\
s.t \quad & y_i(w^T.x_i + b) \geq 1 - \xi_i && \forall\, i = 1, \dots, n \\
& \xi_i \geq 0 && \forall\, i = 1, \dots, n
\end{aligned}
$$

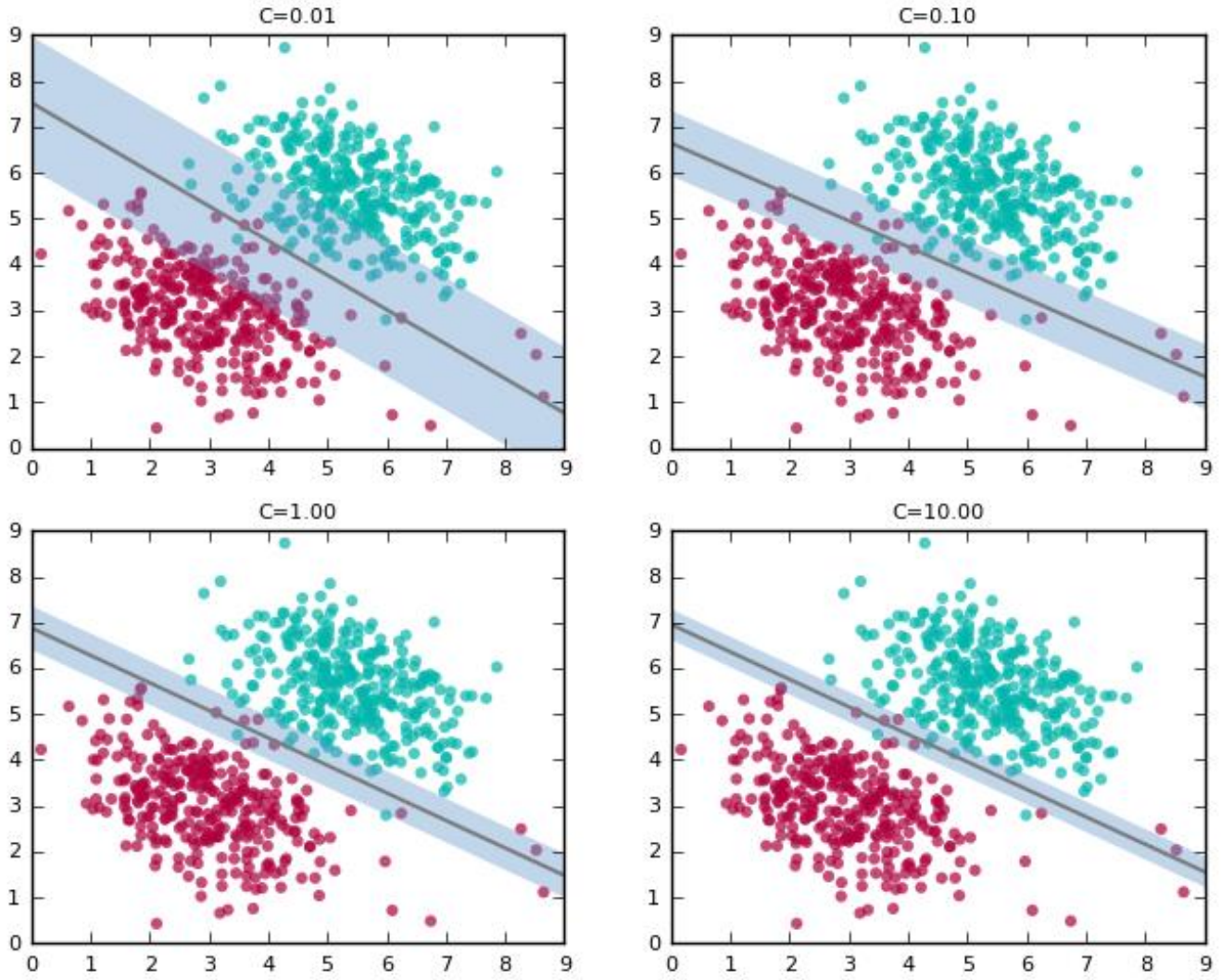
- Chaque contrainte peut être satisfaite si $\boldsymbol{\xi_n}$ est suffisamment grande.
- **C** est un paramètre de régularisation qui gère le compromis entre une marge maximale et le relâchement de la contrainte :
  - Une petite valeur de **C** permet aux contraintes d'être facilement ignorées → **grande marge**
  - Une grande valeur de **C** rend les contraintes difficiles à ignorer → **marge étroite**
- Il s'agit toujours d'un problème d'optimisation quadratique et il existe un minimum unique. Notez qu'il n'y a qu'un seul hyperparamètre*, C.

* Un hyperparamètre est un paramètre dont la valeur est utilisée pour contrôler le processus d'apprentissage. Sa valeur est définie avant de lancer le processus d'apprentissage.

# SVM - Cas linéaire

## Marge poreuse

Illustration de l'influence de l'hyperparamètre C sur la largeur de la marge:



# SVM – Cas non linéaire

## Séparateur non linéaire

**Idée de base** : si vous ne pouvez pas séparer les **positifs** des **négatifs** dans un espace de faible dimension à l'aide d'un hyperplan, projetez alors toutes les données dans un espace de plus grande dimension où vous pouvez les séparer.

Appliquer aux données une transformation $\boldsymbol{\phi}: \mathbb{R}^{\boldsymbol{d}} \rightarrow \mathbb{R}^{\tilde{\boldsymbol{d}}}$ vers un espace de plus grande dimension :

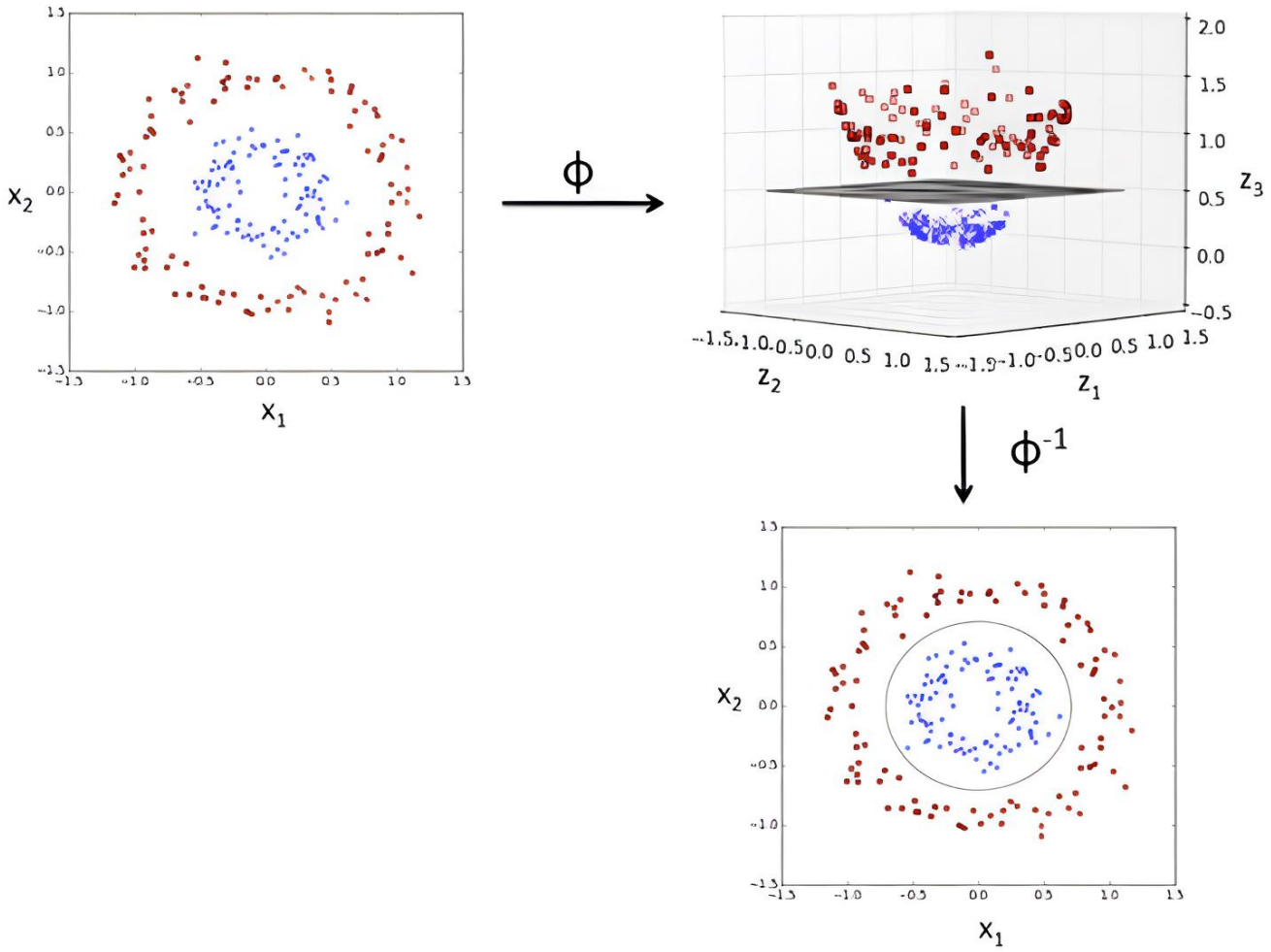
$$z_n = \boldsymbol{\phi}(x_n)$$

Après transformation des données, **on résout le problème de SVM dans ce nouvel espace** de redescription.

$$\begin{array}{lll} min & \frac{1}{2}\tilde{w}^T.\tilde{w} & \\ s.c & y_n(\tilde{w}^T.z_n + b) \geq 1 & \forall\, n = 1, \dots, N \\ & y_n \in \{-1,1\} & \end{array}$$

## Séparateur non linéaire

Ainsi, nous pouvons séparer les deux classes représentées dans le nouvel espace par un hyperplan linéaire qui devient une frontière de décision non linéaire si nous le projetons à nouveau sur l'espace de caractéristiques original.



## Astuce du noyau *(kernel trick)*

**Considérons la transformation suivante:**

$$\phi: \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\phi(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} {x_1}^2 \\ {x_2}^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

**Produit scalaire:**

$$\phi(x).\phi(x') = \begin{pmatrix} {x_1}^2 & {x_2}^2 & \sqrt{2}x_1x_2 \end{pmatrix} \begin{pmatrix} {x'_1}^2 \\ {x'_2}^2 \\ \sqrt{2}x'_1x'_2 \end{pmatrix}$$

$$= {x_1}^2 {x'_1}^2 + {x_2}^2 {x'_2}^2 + 2x_1 x'_1 x_2 x'_2$$

$$= (x_1 x'_1 + x_2 x'_2)^2 = (\boldsymbol{x^T}.\boldsymbol{x'})^2$$

Dans ce cas, la fonction $(\boldsymbol{x^T}.\boldsymbol{x'})^2$ nous permet d'éviter de devoir calculer le produit scalaire de deux exemples de manière explicite, **ce qui nous fait gagner beaucoup de temps.**

Le **but** est de bénéficier de cette astuce dans le cas des SVM, afin de réduire le temps de calcul dans les cas non linéaires.

## Astuce du noyau - formulation duale

Afin de bénéficier de cette astuce, appelée **l'astuce du noyau**, la formulation du SVM doit être reformulée de telle sorte que le produit scalaire apparaisse.

Cela peut être réalisé à l'aide **des multiplicateurs de Lagrange:**

Minimisation d'une fonction quadratique avec une seule contrainte d'inégalité

$$\begin{aligned} \min_{\mathbf{u}\in\mathbb{R}^L} \quad & \tfrac{1}{2}\mathbf{u}^T Q\mathbf{u} + \mathbf{p}^T\mathbf{u} \\ \text{s.c.} \quad & \mathbf{a}^T\mathbf{u} \geq c \end{aligned}$$

**Problème proche**

$$\min_{\mathbf{u}\in\mathbb{R}^L} \quad \tfrac{1}{2}\mathbf{u}^T Q\mathbf{u} + \mathbf{p}^T\mathbf{u} + \max_{\alpha\geq 0} \alpha(c - \mathbf{a}^T\mathbf{u})$$

La variable $\alpha \geq 0$ est appelée multiplicateur de Lagrange.

Dans le cas de la **SVM:**

$$\begin{aligned} \text{Minimiser} \quad & \tfrac{1}{2}\mathbf{w}^T\mathbf{w} \\ s.c. \quad & y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \quad \forall n = 1, \ldots, N \end{aligned}$$

$$\mathcal{L}(b, \mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N} \alpha_n(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))$$

$$\mathcal{L}(b, \mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{w}^T\mathbf{x}_n - b\sum_{n=1}^{N} \alpha_n y_n + \sum_{n=1}^{N} \alpha_n$$

**formulation duale**

# SVM – Cas non linéaire

## Astuce du noyau – formulation duale

$$\mathcal{L}(b, \mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N} \alpha_n (1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))$$

$$\mathcal{L}(b, \mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{w}^T\mathbf{x}_n - b\sum_{n=1}^{N} \alpha_n y_n + \sum_{n=1}^{N} \alpha_n$$

- On a transformé un problème de programmation quadratique **sous contraintes** en... un problème de programmation quadratique **sans contraintes**.
- Cette formulation à un avantage important dans le cas des SVM (cf. Astuce du noyau)

### Interprétation :

- $\boldsymbol{\mathcal{L}}$ doit être minimisé par rapport aux variables primales $(\boldsymbol{b}, \boldsymbol{w})$ et maximisé par rapport aux variables duales $\boldsymbol{\alpha_n}$.
- Si la contrainte est satisfaite, i.e. $\mathbf{1} - \boldsymbol{y_n}\left(\mathbf{w^T x_n} + \mathbf{b}\right) \leq \mathbf{0}$ alors $\boldsymbol{\alpha_n} = \mathbf{0}$, **sauf** pour les vecteurs de support (car en augmentant $\boldsymbol{\alpha_n}$, on minimise $\boldsymbol{\mathcal{L}}$ ce qui est contraire à l'objectif).
- Si la contrainte est violée : $\boldsymbol{\alpha_n}$ va croître pour maximiser, et $\boldsymbol{\mathcal{L}}(b, w)$ va d´décroitre pour que les contraintes soient satisfaites.

# SVM – Cas non linéaire

## Astuce du noyau – minimisation % (b,w)

$$\mathcal{L}(b, \mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N} \alpha_n (1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))$$

$$\mathcal{L}(b, \mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{w}^T\mathbf{x}_n - b\sum_{n=1}^{N} \alpha_n y_n + \sum_{n=1}^{N} \alpha_n$$

**minimisation % (b,w)**

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \quad \Rightarrow \quad \sum_{n=1}^{N} \alpha_n y_n = 0$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$$

**On remplace les valeurs obtenues dans le Lagrangien et on obtient**

$$\mathcal{L}(\alpha) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T \mathbf{x}_m + \sum_{n=1}^{N} \alpha_n$$

**Une formulation à maximiser qui ne dépend que de la variable duale $\alpha_n$**

## Astuce du noyau – formulation duale

**Le problème d'optimisation peut alors s'écrire comme :**

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^N} \quad & \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \Phi(\mathbf{x}_n)^T \Phi(\mathbf{x}_m) - \sum_{n=1}^{N} \alpha_n \\ s.c. \quad & \sum_{n=1}^{N} y_n \alpha_n = 0 \\ & \alpha_n \geq 0 \end{aligned}$$

**On a donc la fonction de décision finale suivante :**

$$g(\mathbf{x}) = signe\left( \sum_{\alpha_n^* > 0} y_n \alpha_n^* \Phi(\mathbf{x}_n)^T \Phi(\mathbf{x}) + b^* \right)$$

- Seuls les vecteurs de support sont nécessaires pour prédire la classe d'une nouvelle donnée, car ils définissent l'hyperplan séparateur.
- Le calcul des $\boldsymbol{\alpha_n}$ se fait à l'aide d'un solveur.
- Cette nouvelle formulation permet de bénéficier de **l'astuce du noyau**.

## Astuce du noyau – formulation duale

Dans la formulation précédente (pour l'apprentissage et la décision), le seul calcul qui est réalisé dans l'espace de redescription est un produit scalaire.

$$\boldsymbol{\phi}(\boldsymbol{x}).\boldsymbol{\phi}(\boldsymbol{x}')$$

L'idée est trouver une fonction qui calcule conjointement la transformation et le produit scalaire.

$$\boldsymbol{K}_{\boldsymbol{\phi}}(\boldsymbol{x}.\boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}).\boldsymbol{\phi}(\boldsymbol{x}')$$

$\boldsymbol{K}_{\boldsymbol{\phi}}$ est une fonction noyau.

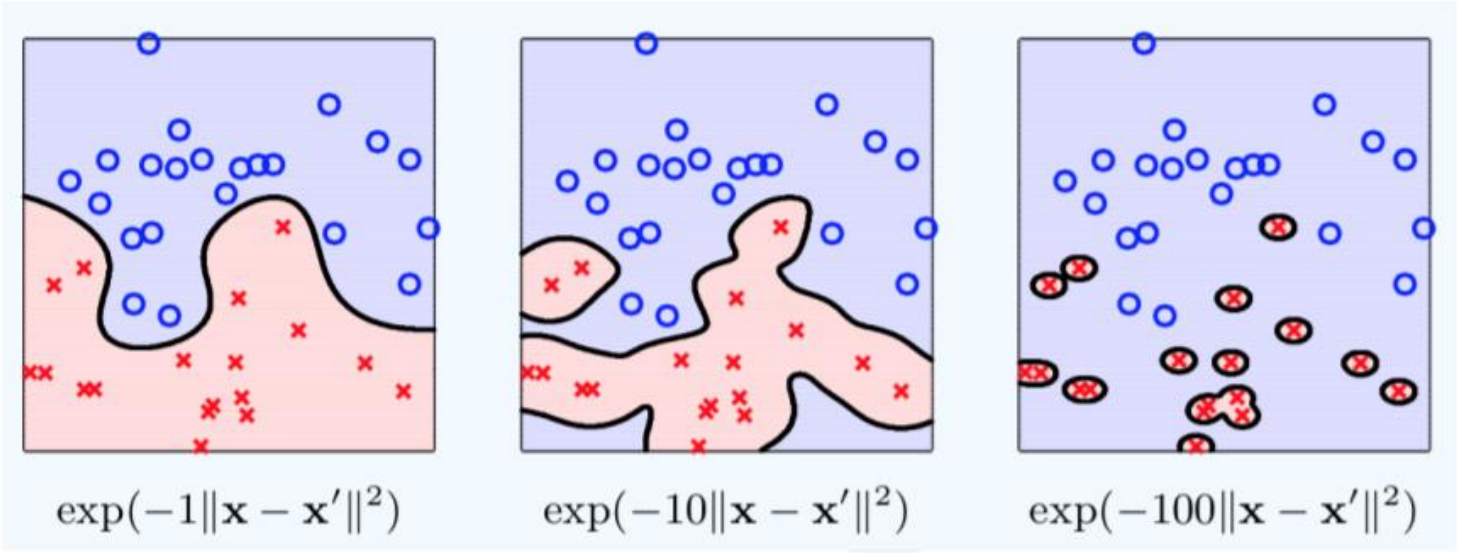
On connaît un certain nombre de fonctions noyaux, parmi lesquelles :

- Le noyau polynomial, $K(x, x') = (\alpha x^\top \cdot x' + \lambda)^d$
- Le noyau gaussien, $K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$
- Le noyau laplacien, $K(x, x') = \exp\left(-\frac{\|x-x'\|}{\sigma}\right)$
- Le noyau rationnel, $K(x, x') = 1 - \frac{\|x'-x\|^2}{\|x'-x\|^2+\sigma}$

## Astuce du noyau – Noyau gaussien

Illustration de classification par noyau gaussien

- Noyau gaussien : $K(\mathbf{x}, \mathbf{x}') = \exp(\frac{-||\mathbf{x}-\mathbf{x}'||}{2\sigma^2})^2$
  - projection dans un espace de dimension infinie



## Conclusion

- Ne permet pas l'extraction d'un modelé compréhensible/interprétable.
- Inadapté a la fouille de très grands volumes de données.
- Choix du noyau : pas de solution a l'heure actuelle si ce n'est par essai/erreur.
- Fournit un optimum global, mais qu'est ce que cela signifie vraiment pour un jeu de données réel ?
- Pour faire de la fouille de données, il faut comprendre l'algorithme que l’on utilise. Clairement, une bonne compréhension des SVM n'est pas aisée.

## Application

L'**objectif** de la 3ème application est de :

1. **Créer** un modèle prédictif **SVM** à partir du jeu d'apprentissage,
2. **Valider** les performances prédictives du modèle à l'aide du jeu de validation.
3. **Evaluer** la généralisation du modèle à l'aide du jeu de test.