Demi-Journée SYDRE: Formation "Machine Learning"

Introduction à l'apprentissage automatique

Alexis Lechervy
<alexis.lechervy@unicaen.fr>

10 juin 2022

Sommaire

Présentation de la journée

L'orateur

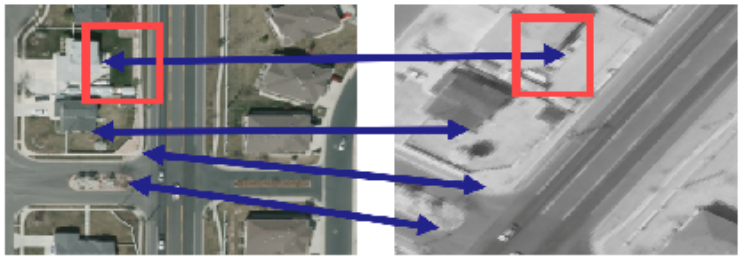
Alexis Lechervy <alexis.lechervy@unicaen.fr> : Maitre de conférence à l'université de Caen.
Thème de recherche :

- Apprentissage automatique,
- Apprentissage de fonction de similarité,
- Apprentissage avec de données multimodales,
- Apprentissage par DeepLearning, Boosting et méthode à noyau...

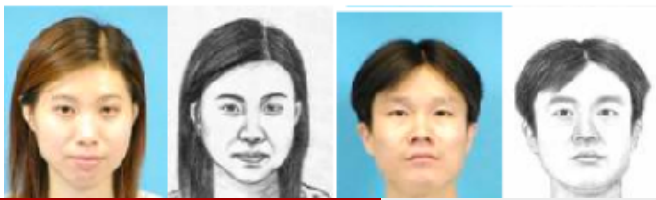
Exemples d'application :

Adaptation de domaine

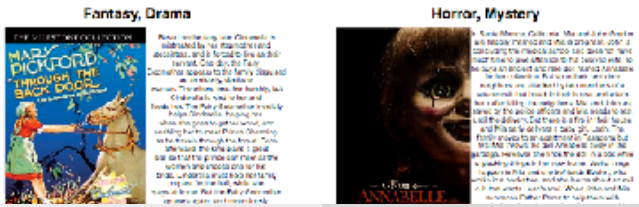
Détection de patch



Reconnaissance photos/dessin



Catégorisation de film



Présentation de la journée

Matinée (9h30-12h30) : Introduction à l'apprentissage automatique

1. Introduction générale,
2. Des données à de l'apprentissage,
3. Quelques algorithmes de classifications.

Sommaire

Apprendre ? Qu'est ce que c'est ?

Apprendre c'est s'adapter.

- **Apprendre c'est s'adapter** à des situations **nouvelles et inconnues** en prenant en compte l'expérience passée.
- Apprendre est une **propriété** humaine **essentielle**.
- Apprendre signifie**"s'améliorer afin d'être meilleur"** (selon un critère donné) lorsqu'une situation similaire se présente.

Apprendre ce n'est pas du "par-coeur".

- Ne pas confondre l'apprentissage et la **récitation par-coeur**.
- N'importe quel ordinateur peut réciter "par-coeur", la difficulté est de généraliser à des situations nouvelles et inconnues.

Mais pourquoi apprendre à un ordinateur ?

- Pouvoir gérer une quantité de données très importante de manière automatique.
- Pouvoir effectuer une action dans un contexte non prévu préalablement sans l'intervention d'un humain.
- Pouvoir prévoir des comportements ou des évolutions pour aider à la prise de décision.

L'Apprentissage en informatique, quand faut-il l'utiliser ?

Quand l'utiliser ?

- L'expertise humaine n'est pas possible (ex : navigation sur Mars)
- La quantité d'information est trop grande pour être traitée par un humain (ex : recherche d'une personne dans une base d'image)
- Besoin de traitement temps réel (ex : mise au point d'un appareil photo sur un visage)
- Automatisation d'une chaine de traitement (ex : détection d'anomalie sur une chaine de montage)
- Les êtres humains ne savent pas expliquer leurs expertises (ex : reconnaissance de la parole)
- Trouver une solution optimale à un problème (ex : trouver le meilleur modèle)

Quand ne pas l'utiliser ?

- Pour réaliser une application simple dont le comportement est facilement définissable,
- Pour réaliser une tache impossible,
- Lorsque les données sont mal définis notamment si elle contient des biais dû à des préjugés ou des stéréotypes.
- Pour confirmer ou invalidité l'avis des experts sans discernement.

$\Longrightarrow$ l'apprentissage se fait à partir de données, l'algorithme apprendra et reproduira les défauts qu'elles peuvent contenir.

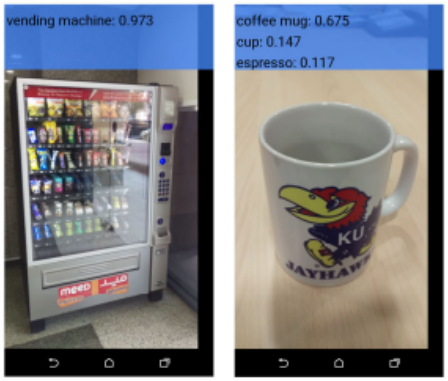
Des exemples d'applications

Navigation autonome

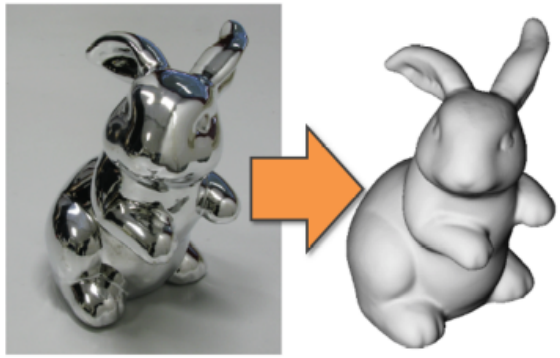
Recommandation de contenu

Analyse de jeu de Go

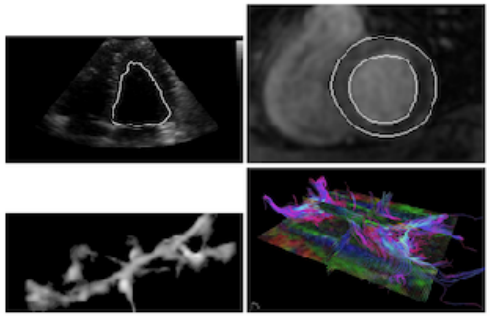
Reconnaissance de contenu



Reconstruction 3D



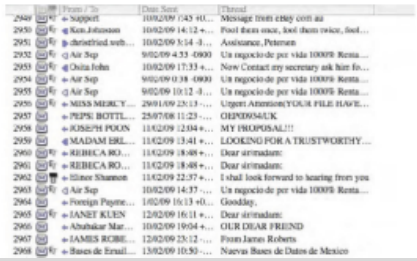
Aide au diagnostique



Détection et suivi d'objet

Transfert de style

Filtrage de mail



Sommaire

Définitions

Définition intuitive

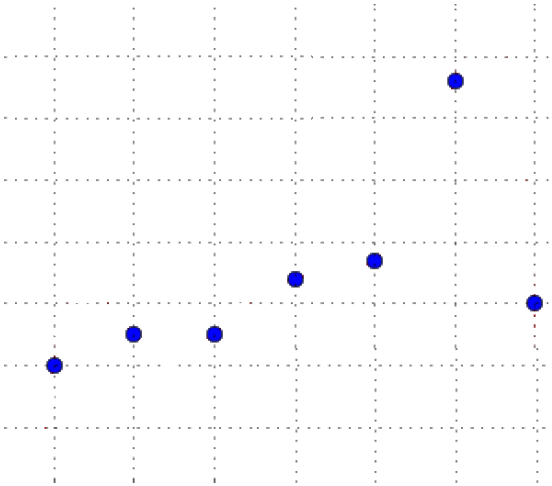
L'apprentissage automatique consiste en la conception et le développement d'algorithmes qui permettent aux ordinateurs (machines) d'améliorer leurs performances au fil du temps sur une base de données.

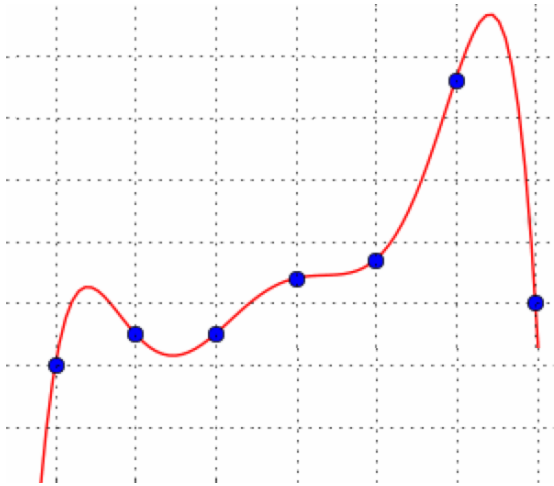
Définition formelle (Mitchell, 1998)

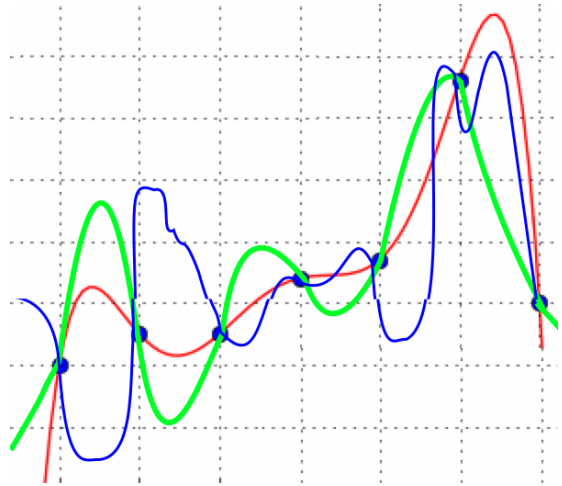
A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.
(Machine Learning de Tom M. Mitchell, Chapitre 1, page 2).

Pourquoi est ce difficile ?

- Etant donné un **nombre fini de données d'apprentissage**, il faut en déduire une **relation pour un domaine infini**.
- Problème : il y un **nombre infini** de telles **relations**.
- Comment définir cette relation ?
- Quelle relation est la plus appropriée ?







Alors comment faire ?

Principe du rasoir d'Occam

William of Occam (un Moine du 14$^{\text{ième}}$ siècle) propose d'appliquer le **principe de parcimonie** :

> *On ne devrait pas augmenter, au-delà de ce qui est nécessaire, le nombre d'entités requises pour expliquer quoi que ce soit.*

En d'autre mot : Lorsque de nombreuses solutions sont disponibles pour un problème donné, nous devons sélectionner **la plus simple**.

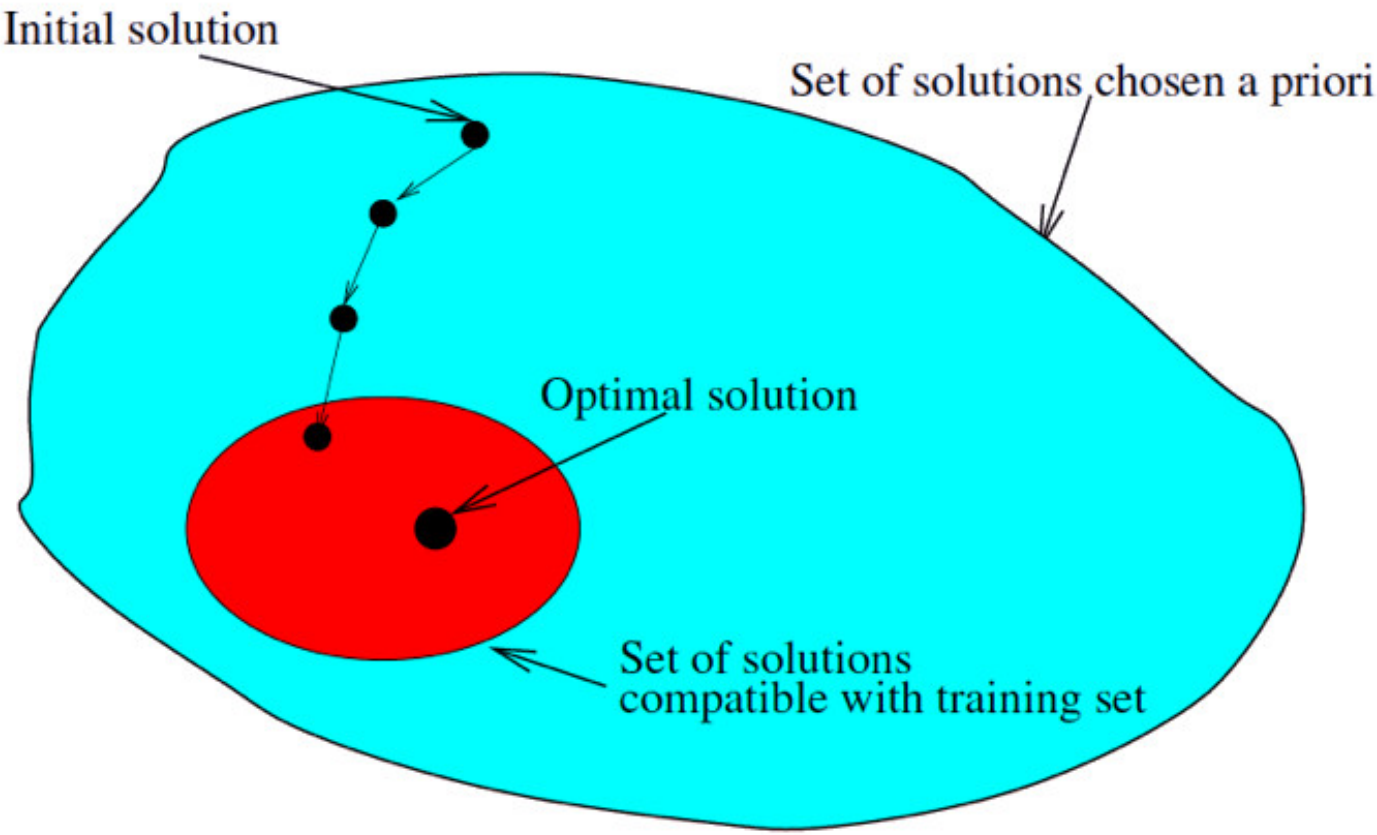
$\Longrightarrow$ **Mais que signifie la plus simple dans notre cas ?**

Comment définir la simplicité d'une solution ?

- Nous devons utiliser des connaissances apriori du problème à résoudre pour définir ce qui est une solution simple.
- Exemples d'apriori : solution lisse, solution à peu de coefficients...

L'apprentissage : une recherche d'une solution parmi un ensemble de possibilité.

L'apprentissage consiste à choisir une solution parmi un ensemble de solution défini selon un apriori. On recherche la solution optimale. C'est à dire la solution "la plus simple" qui répond à notre problème.



Un peu d'histoire.

Histoire

- 1960 : IA (Intelligence Artificielle) symbolique, les ordinateurs apprennent des règles à partir des données mais une analyse des statistiques sous-jacentes est rarement faite.
- Fin 1960, début 1970 : introduction de modèles de discrimination linéaires dont le fameux perceptron de Minsky et Papert, 1969.
- 1980 : Introduction des réseaux de neurones artificiels.
- 1990-2000 : ère de l'apprentissage statistique (méthodes à noyaux, SVM, arbres de décisions, boosting, modèles graphiques).
- 2010- : ère du BigData, du BigLearning et Deep Learning.

Pourquoi l'apprentissage a mis tant de temps à se développer ?

- Des ordinateurs plus rapides avec une plus grande mémoire pour représenter des modèles sont désormais disponibles.
- Les méthodes d'analyse numériques sont disponibles sur l'ordinateur de bureau.
- De grandes quantités de données disponibles qui peuvent être exploitées (profils utilisateurs, flickr, réseaux sociaux...).
- De nombreux ensembles de données avec des jeux de données partiels / incomplets.

Sommaire

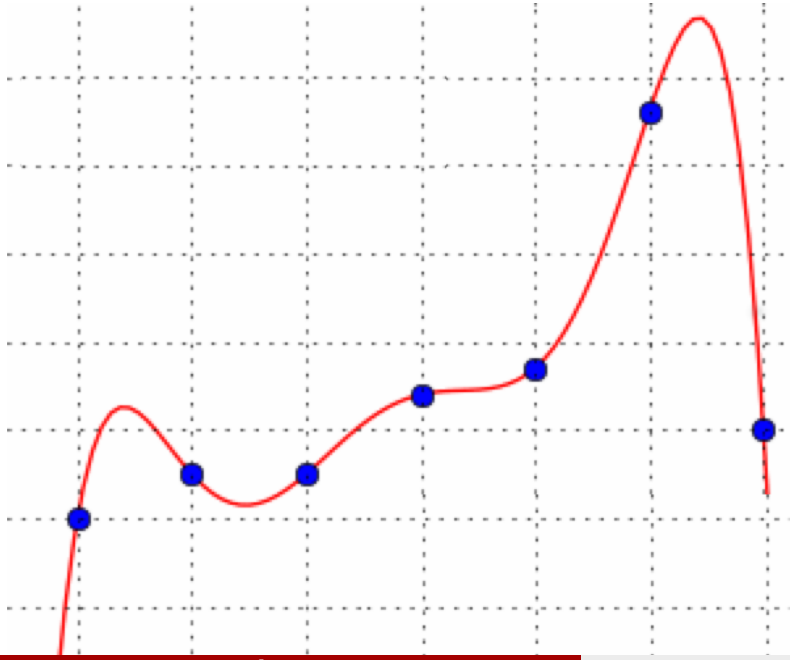
La régression

Principes

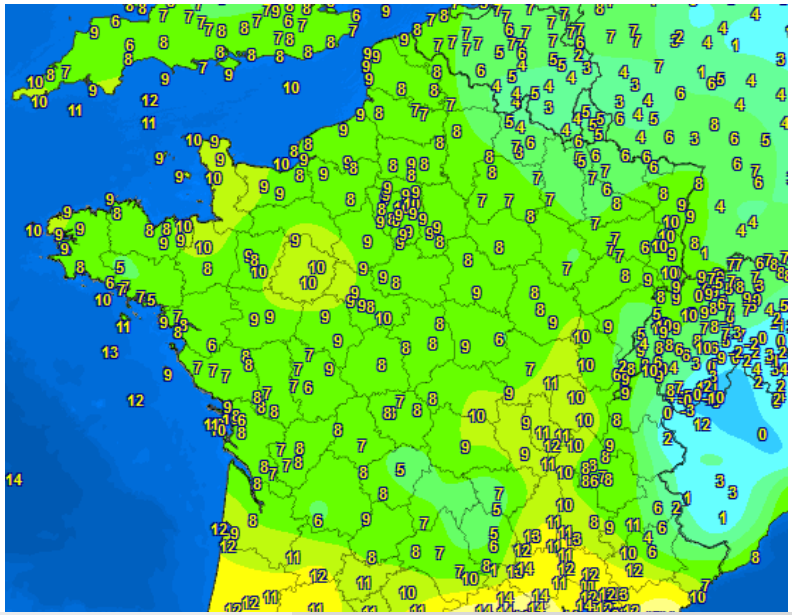
- Trouver la relation entre une variable et une ou plusieurs autres variables.
- Trouver la fonction qui minimise un critère d'erreur entre les valeurs de la fonction aux points d'apprentissage et les observations.
- Recherche d'une fonction de type $y = f(x, a, b) = (ax + b)$.

Exemples d'applications

Retrouver la courbe passant par un ensemble de points.



Faire une carte de température en fonction de points de prise de mesures :

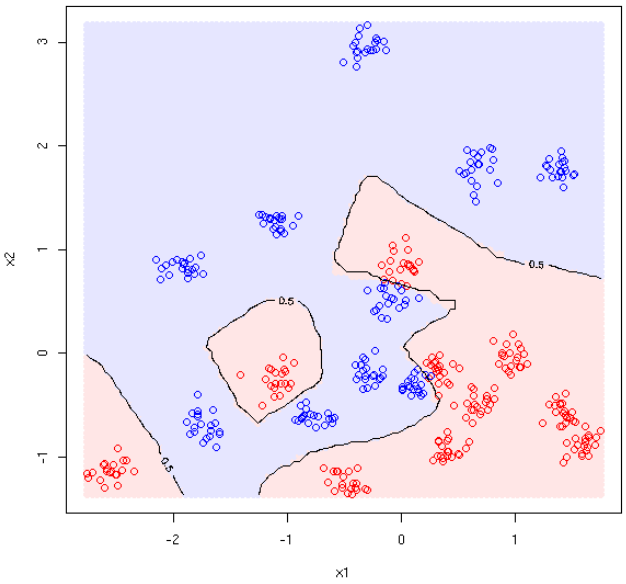


La classification

Principes

- Attribuer une classe à chaque objet.
- Utilise des données statistiques sur les objets pour choisir la classe.
- Recherche d'une fonction $y = f(x, a, b) = \text{sign}(ax + b)$

Exemples

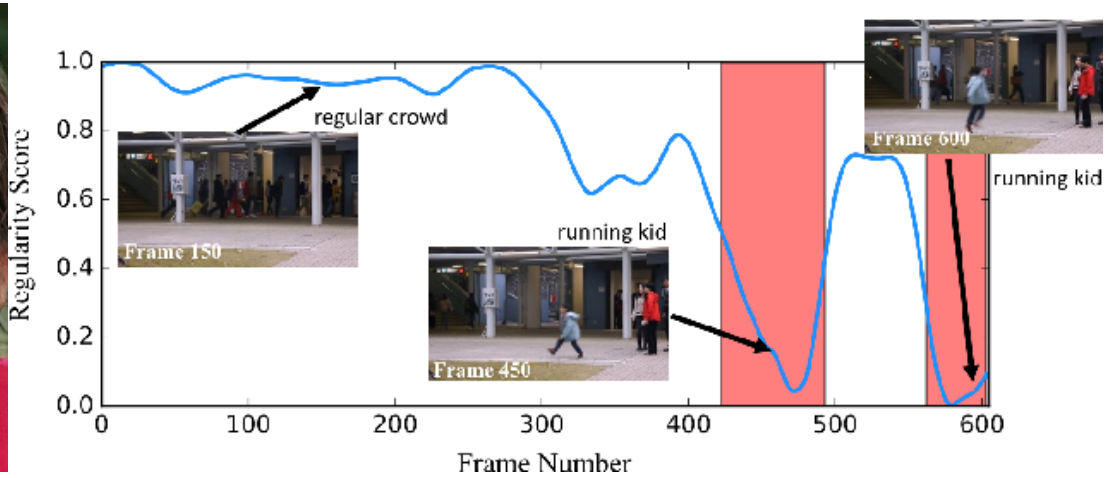


Estimation de densité

Principes

- Trouver les paramètres d'une loi de probabilité permettant d'estimer au mieux une distribution de points.

Exemples

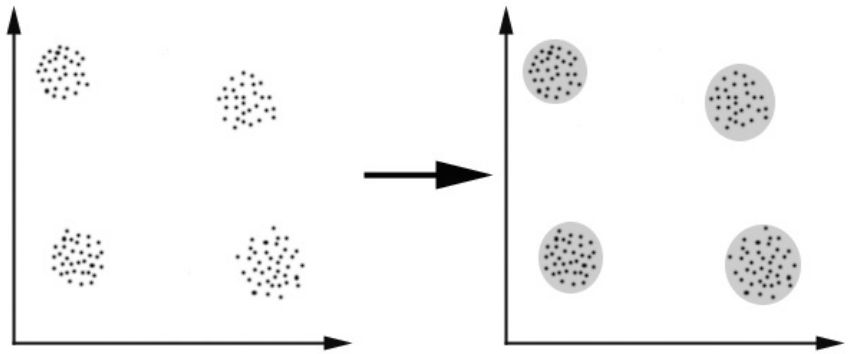


L'apprentissage non supervisé (Clustering)

Principes

- Diviser les données en plusieurs groupes séparés,
- Extraire une connaissance organisée sans intervention humaine,
- les données les plus similaires sont associées au sein d'un groupe homogène
- les données considérées comme différentes se retrouvent dans d'autres groupes distincts
- Pas d'apriori sur les données
- Il y a une seule entrée, les données collectées

Exemple de clustering

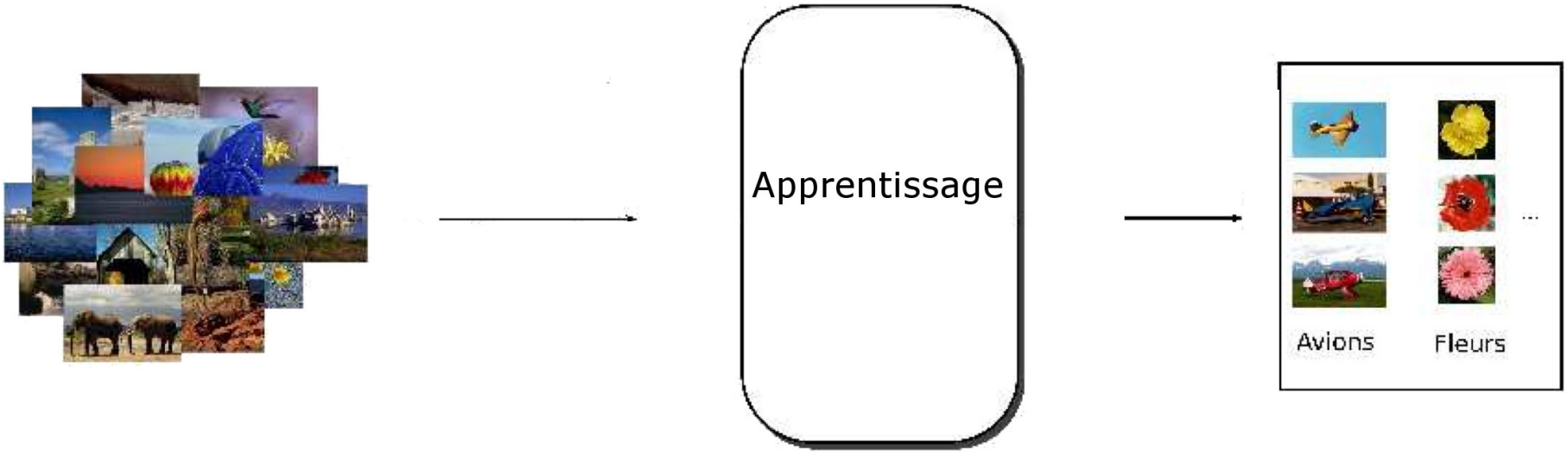


L'apprentissage supervisé

Principes

- On détermine automatiquement une règle à partir de données d'apprentissage annotées par un expert,
- Un expert a défini un ensemble de couples (donnée,label),
- Il y a un apriori sur les données,
- Les données entrées sont des couples (données collectées, observations).

Exemple : Catégorisation d'image



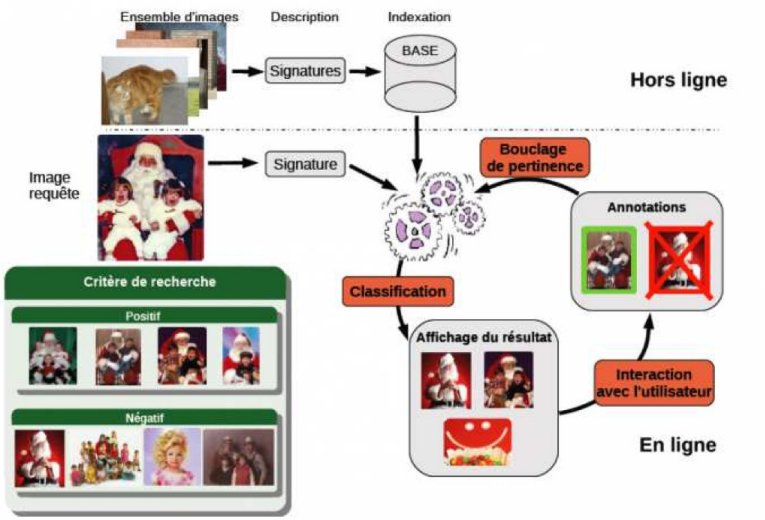
L'apprentissage semi-supervisé

Principes

- On dispose de quelques exemples labellisés.
- Les autres ne le sont pas.
- Permet de travailler avec moins de labels.

Exemples d'apprentissage semi-supervisé

- L'apprentissage interactif.
- L'apprentissage sur des ensembles de données trop important.



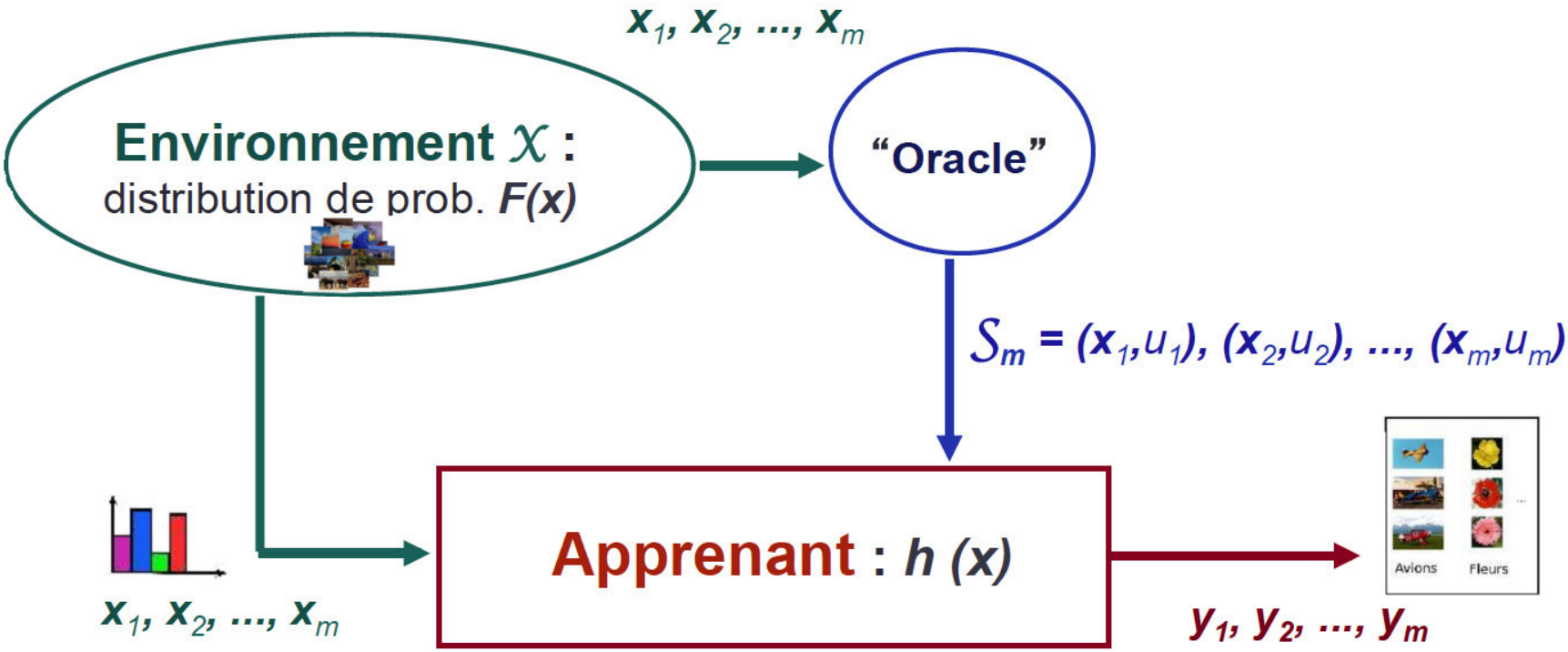
Sommaire

Contexte de ce cours

Problème étudié

Nous aborderons aujourd'hui uniquement des cas de **classification supervisée**.

Déroulement classique de l'apprentissage supervisé



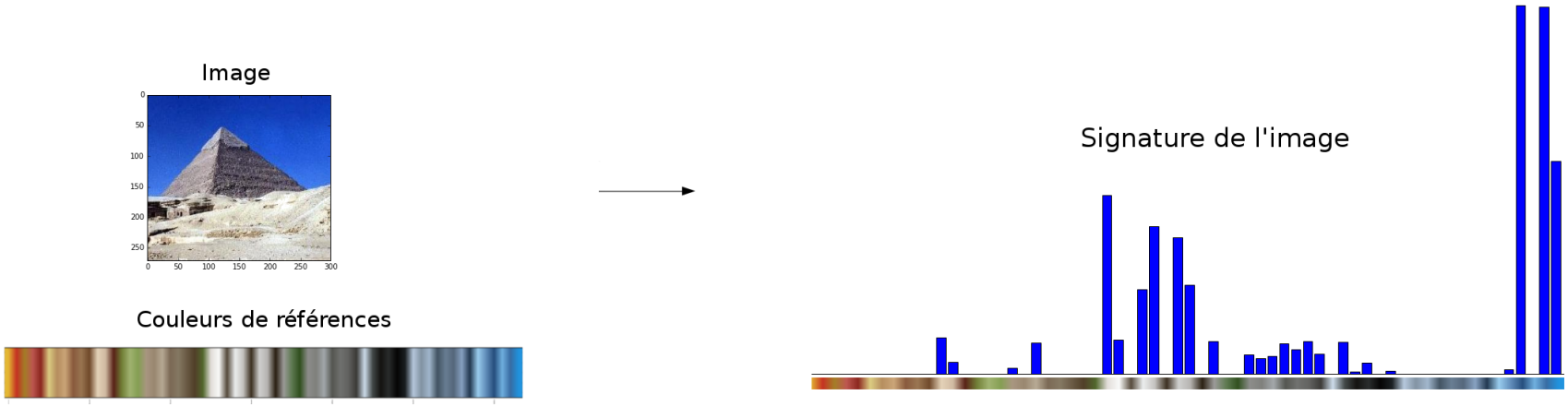
Induction : Proposer des **lois générales** à partir de l'observation de **cas particuliers**.

Étapes classiques d'un apprentissage

Étapes

1. Calcul d'un vecteur de caractéristique à partir des données. Généralement en utilisant un algorithme conçu en fonction de la nature des données.
2. Apprentissage de la tâche cible à partir des vecteurs précédents et d'un apriori. Généralement, on cherche à produire une fonction paramétrique définissant la frontière de décision, en minimisant un certain critère d'erreur.

Exemple de vecteur de caractéristique

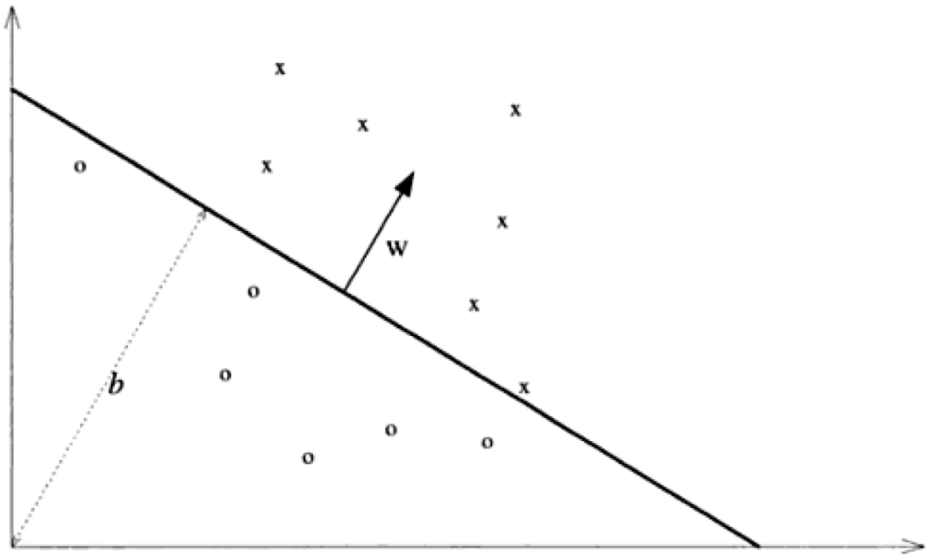


Étapes classiques d'un apprentissage

Étapes

1. Calcul d'un vecteur de caractéristique à partir des données. Généralement en utilisant un algorithme conçu en fonction de la nature des données.
2. Apprentissage de la tâche cible à partir des vecteurs précédents et d'un apriori. Généralement, on cherche à produire une fonction paramétrique définissant la frontière de décision, en minimisant un certain critère d'erreur.

Exemple de fonction paramétrique



$$f_w(x) = \begin{cases} 1 \text{ si } \langle w, x \rangle + b > 0 \\ -1 \text{ sinon} \end{cases}$$

Étapes classiques d'un apprentissage

Étapes

1. Calcul d'un vecteur de caractéristique à partir des données. Généralement en utilisant un algorithme conçu en fonction de la nature des données.
2. Apprentissage de la tâche cible à partir des vecteurs précédents et d'un apriori. Généralement, on cherche à produire une fonction paramétrique définissant la frontière de décision, en minimisant un certain critère d'erreur.

Exemple de fonction de cout

On peut chercher à maximiser la probabilité d'être bien classé en utilisant par exemple la cross-entropie.

Résumé

Ingrédients d'une méthode d'*Apprentissage automatique*

- Un espace de représentation des données en **vecteurs caractéristiques**.
- Une famille de **fonctions paramétriques** définissant la frontière de décision.
- Un **critère d'erreur** permettant les valeurs optimaux des paramètres de la fonction paramétrique pour les données d'apprentissage.

Recette

Pour apprendre/trouver les meilleurs paramètres de la fonction paramétrique, il faut utiliser un algorithme de résolution de problème d'optimisation.
En fonction du problème d'optimisation et de la résolution de ce dernier, on obtient une méthode d'apprentissage différente : Perceptron, SVM, Classifieur logistique, Adaboost, Réseau de neurone...

Les Hyper-paramètres

Les hyper-paramètres sont des paramètres fixé avant l'apprentissage. Contrairement aux paramètres de la fonction apprise, leurs choix ne découle pas directement de l'algorithme d'apprentissage.
Le choix des hyper-paramètres se fait généralement en fonction des performances de la fonction apprise sur un ensemble de validation.

Sommaire

Sommaire

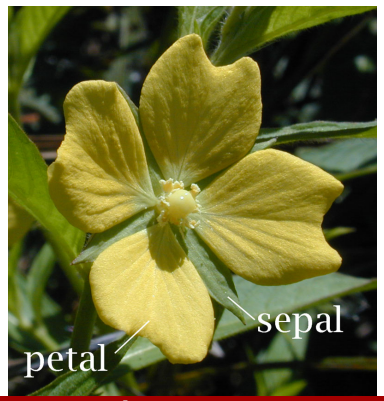
Vecteur de caractéristique simple

Vecteur simple

Certaines bases de données sont directement sous la forme d'un vecteur par exemple d'apprentissage. Elles ne nécessitent pas de traitement particulier et peuvent être traité directement par les algorithmes d'apprentissage automatique.

Exemple : la base d'Iris de Fisher

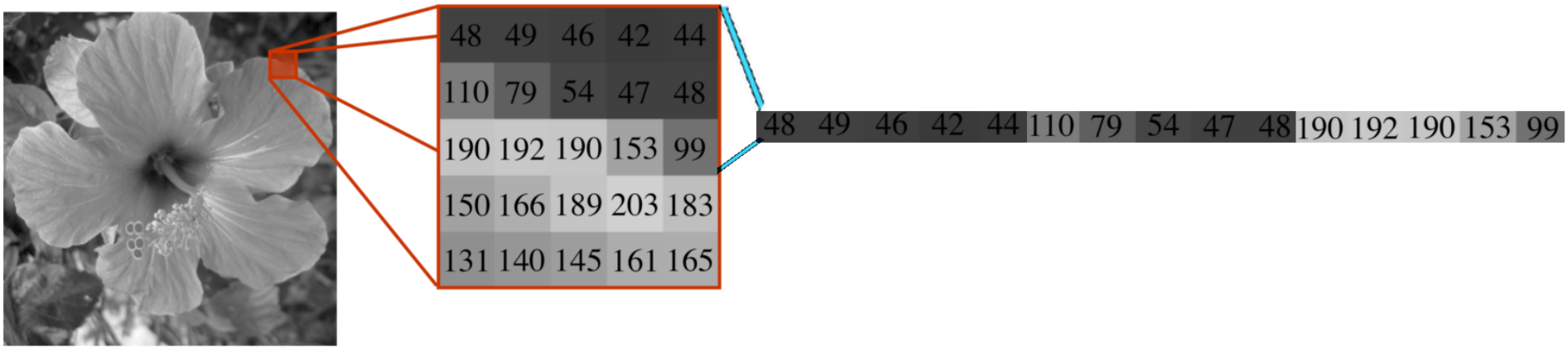
Les iris de Fisher `https://archive.ics.uci.edu/ml/datasets/iris` est une base classique d'apprentissage automatique proposé par Ronald Fisher en 1936. Elle comprend 150 échantillons d'iris répartit équitablement selon 3 catégories. Pour chaque fleur, il est mesuré 4 attributs : longueurs et largeur du pétale et du sépale :



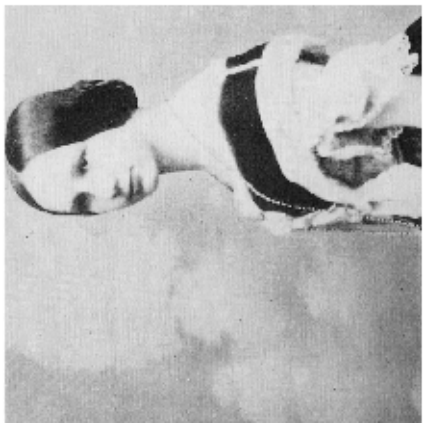
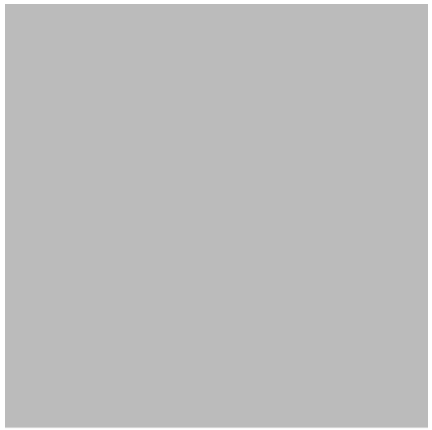
longueur des sépales (en cm)	largeur des sépales (en cm)	longueur des pétales (en cm)	largeur des pétales (en cm)	Espèce
5.0	2.0	3.5	1.0	*I. versicolor*
6.0	2.2	5.0	1.5	*I. virginica*
4.5	2.3	1.3	0.3	*I. setosa*
5.5	2.3	4.0	1.3	*I. versicolor*
6.3	2.3	4.4	1.3	*I. versicolor*
5.0	2.3	3.3	1.0	*I. versicolor*
4.9	2.4	3.3	1.0	*I. versicolor*
5.5	2.4	3.8	1.1	*I. versicolor*
5.1	2.5	3.0	1.1	*I. versicolor*
4.9	2.5	4.5	1.7	*I. virginica*
6.7	2.5	5.8	1.8	*I. virginica*
5.7	2.5	5.0	2.0	*I. virginica*

Une image vu comme un vecteur de pixel ?

L'idée



Mais est-ce une bonne idée ?

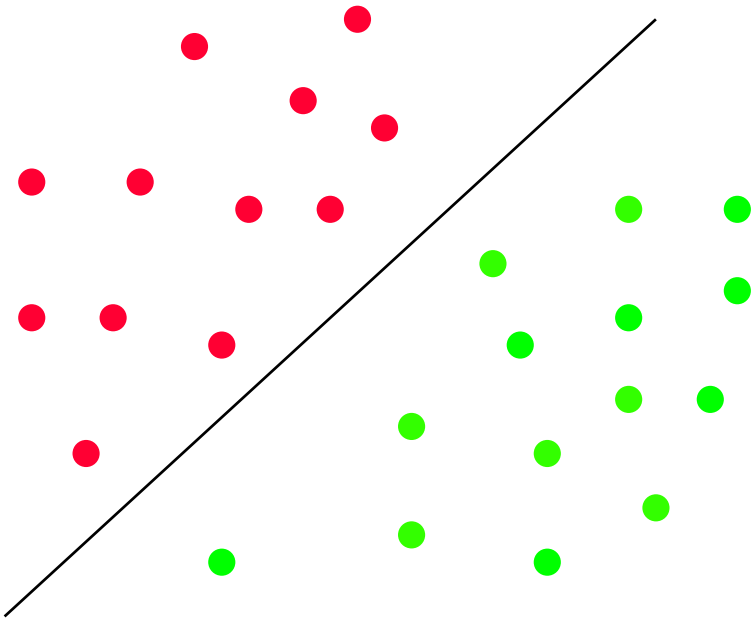




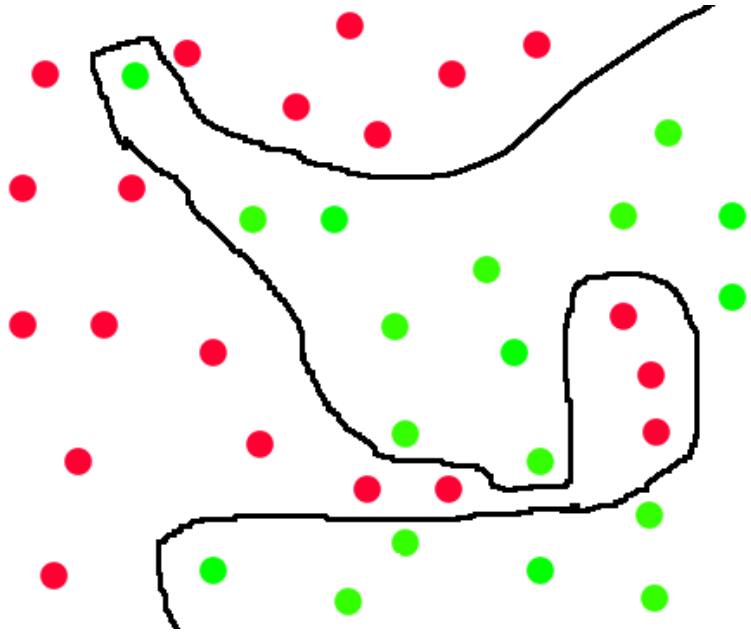
Distance image 1 / image 2 : 174.702, image 1 / image 3 : 119.80

Une image vu comme un vecteur de pixel ?

Rappels

1. L'apprentissage consiste à trouver une frontière de séparation entre les classes dans l'espace de représentation des données traitées.
2. Le principe de parcimonie nous conseille de toujours privilégié la "solution la plus simple".





$\Longrightarrow$ Il ne semble donc pas nécessairement pertinent de voir une image comme un simple vecteur de pixel.

Sommaire

Comment construire un descripteur pour les images

L'idée

On va chercher à construire un vecteur de propriétés statistiques élémentaires des pixels avec des propriétés d'invariances (par exemple invariance à la translation ou la rotation).

Exemples

- La proportion de couleurs présentent (par exemple le rouge peut être très significatif pour trouver des pommes ou des Ferraris),
- La proportion de formes caractéristiques,
- La quantité de certaines textures...

Ils existent de nombres signatures dans la littérature. Il n'y a pas une optimale, tout dépend du problème que l'on cherche à résoudre.

Les signatures basés contours

Principes

- Faire des histogrammes sur des primitives de formes.
- Il faut savoir extraire des contours, approximer des formes par des polygones, calculer des distances sur des graphes...

Limites

- Généralement pas invariant à de fortes rotations ou à des changements d'échelles importants.
- Il est généralement difficile de séparer les objets d'intérêt et le fond.

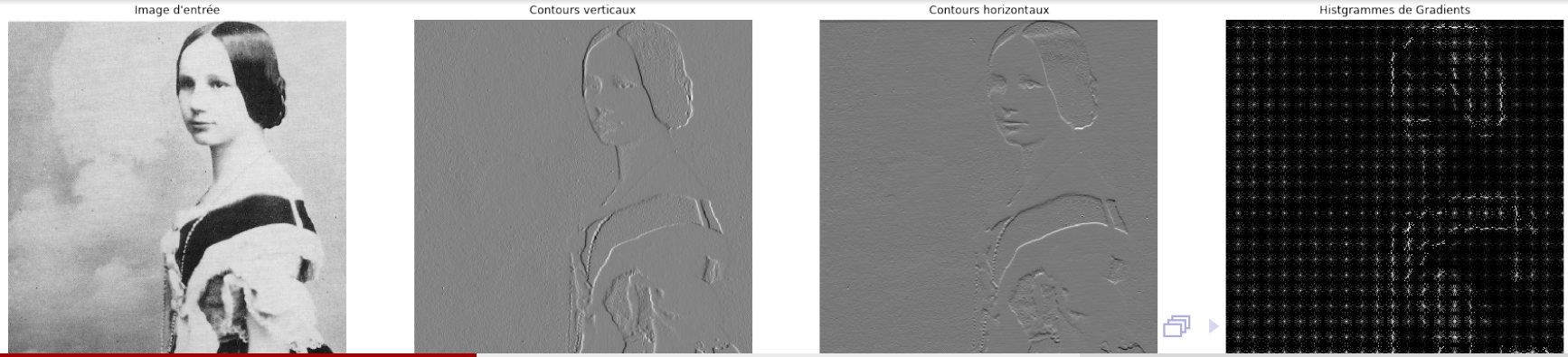
Les histogrammes de gradient orienté (HOG)

Principes

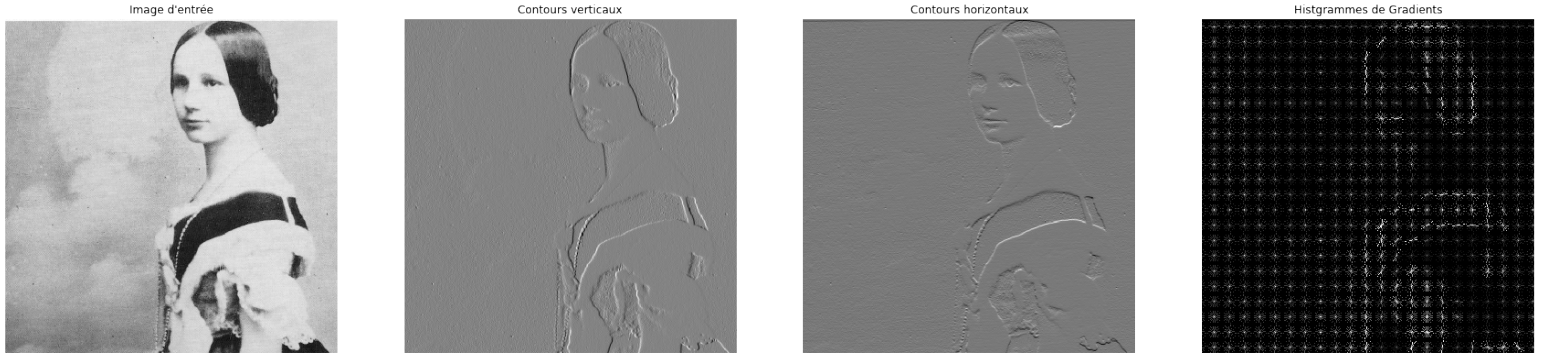
- Calculer un histogramme des gradients de l'image selon un nombre donnée d'orientation.

Étapes

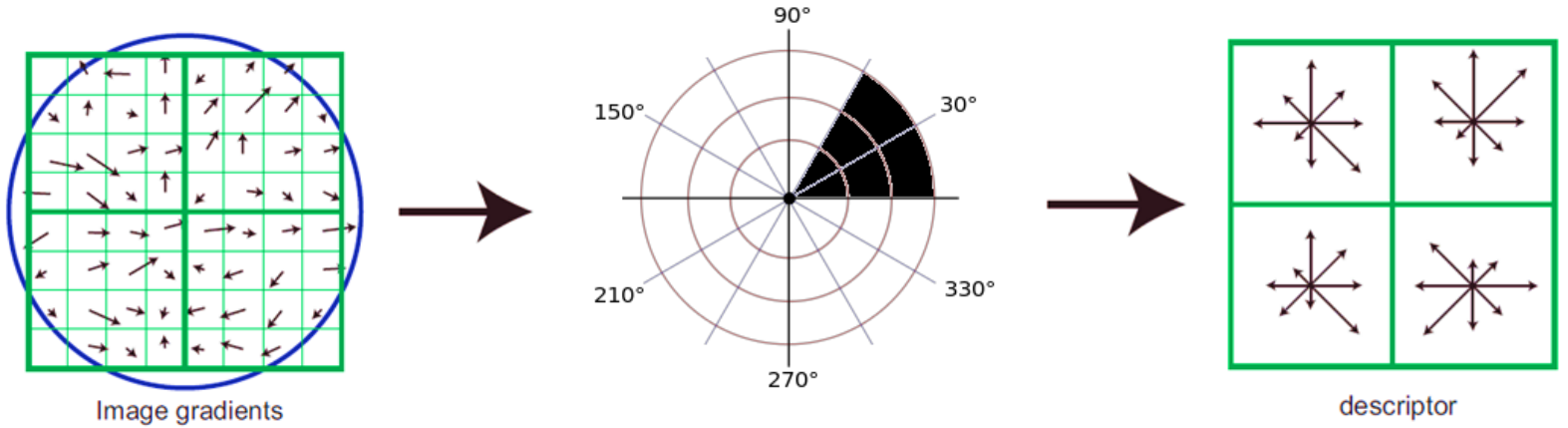
1. Calculez le gradient de l'image. Plusieurs méthodes sont possibles. Par exemple appliquez un filtre dérivatif horizontale $[-1, 0, 1]$ et vertical $[-1, 0, 1]^\top$ ou appliqué un filtre de Sobel.
2. Construction d'un histogramme des gradients dans des cellules de plusieurs pixels. Chaque pixel vote pour l'orientation qui lui est la plus proche, pondéré par l'intensité du gradient en ce point.



Les histogrammes de gradient orienté (HOG)



$$\begin{matrix} 124 & 220 & 220 & 245 \\ 123 & 221 & 190 & 243 \\ 134 & 200 & 199 & 205 \\ 99 & 90 & 180 & 123 \end{matrix} \Longrightarrow \begin{matrix} . & -76 & -25 & . \\ . & -67 & -22 & . \\ . & -65 & -5 & . \\ . & -81 & -33 & . \end{matrix} , \begin{matrix} . & . & . & . \\ -10 & 20 & 21 & 40 \\ 24 & 131 & 10 & 120 \\ . & . & . & . \end{matrix} \Longrightarrow \begin{matrix} . & . & . & . \\ . & (-67,20) & (-22,21) & . \\ . & (-65,131) & (-5,10) & . \\ . & . & . & . \end{matrix}$$



Exemple de reconnaissance de voiture sur la base VOC

Sommaire

Descripteurs pour le texte

Objectif

- Construire un vecteur caractéristique représentant des propriétés statistiques sur les mots du textes.

Du texte au descripteur

- Idée : on peut compter le nombre d'occurence des mots du textes.
- On définit un vecteur dont la taille est celle du dictionnaire de la langue du texte cible et dont les valeurs sont le nombre de fois qu'apparait chaque mots du texte.
- On peut utiliser un dictionnaire regroupant les mots de la même famille, utilisant uniquement la racine des mots et ignorant les mots les plus courant comme "le, la, un, est...".

Exemple

Maître Corbeau, sur un arbre perché,
Tenait en son bec un fromage.
Maître Renard, par l'odeur alléchée...

Dictionnaire :{Maitre, Corbeau, sur, un, arbre, percher, tenir, bec, fromage, renard, par, odeur, allecher}

Vecteur :[2,1,2,1,1,1,1,1,1,1,1,1]

Variantes du nombre d'occurence des termes du dictionnaire

Variantes

- Binaires : 0,1 en fonction de la présence ou l'absence du mots,
- Fréquence : $\frac{\text{nombre d'occurence}}{\text{nombre de mots du texte}}$
- Normalisation logarithmique de la fréquence : $1 + \log\left(\frac{\text{nombre d'occurence}}{\text{nombre de mots du texte}}\right)$
- Normalisation par le max : $K + (1 - K)\frac{\text{fréquence du mot}}{\text{la fréquence du mot le plus fréquent}}$, K étant un paramètre.

TF-IDF

Principes

- Les termes peut fréquent du vocabulaire sont souvent plus discrimant que les termes les plus courant.
- On va pondéré la fréquence des mots par un poid prenant en compte la "rareté" du mots.

Term frequency (TF)

Fréquence : $tf = \frac{\text{nombre d'occurence}}{\text{nombre de mots du texte}}$

Inverse document frequency (IdF)

La fréquence inverse de documenest une mesure de l'importance du terme dans l'ensemble du corpus : $idf = \log \frac{\text{nombre total de documents dans le corpus}}{\text{nombre de documents où le terme apparaît}}$

Calcul du Tf-Idf

$TF\_IdF = tf \times idf$

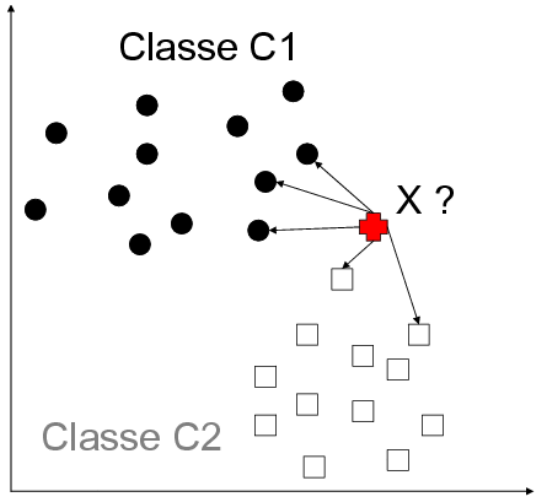
Sommaire

Sommaire

Les k-plus-proche voisins (kPPV) | k-nearest neighbor (kNN)

Principe

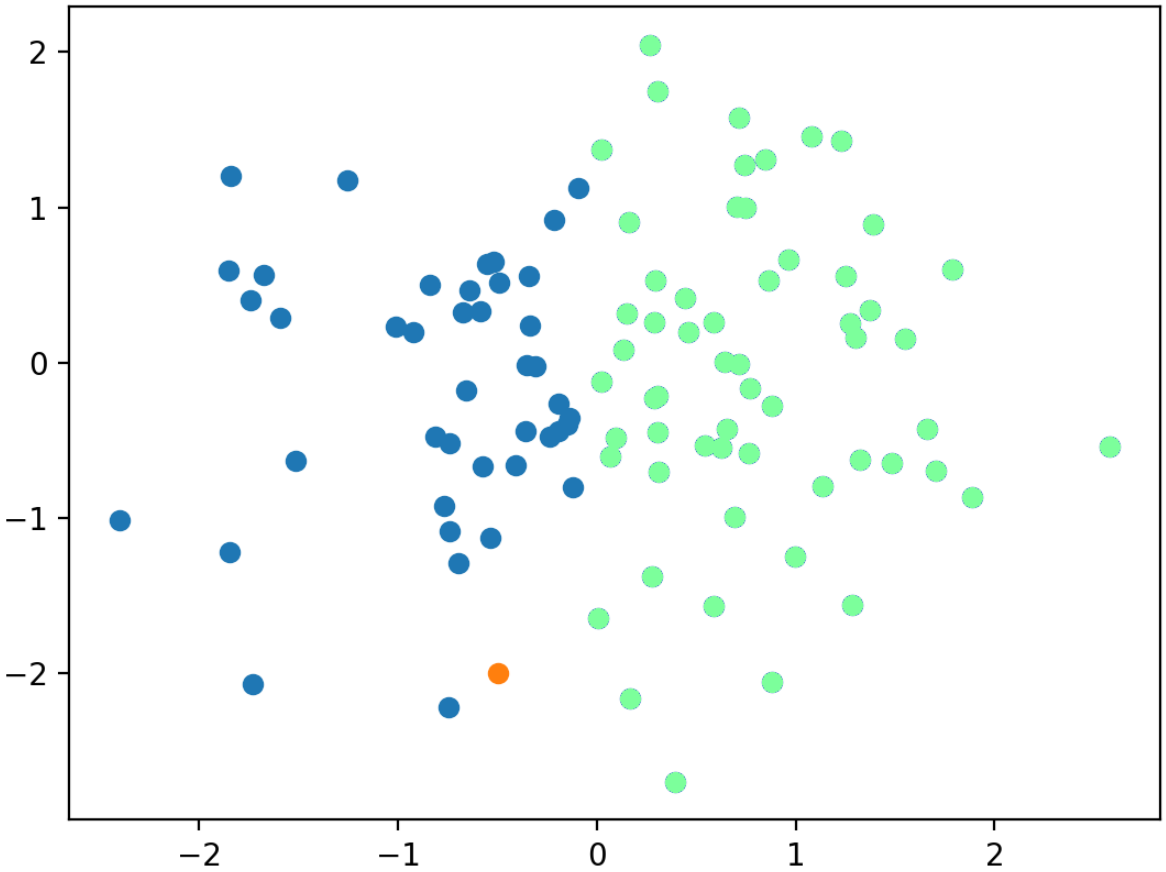
- La classe d'un exemple est un vote majoritaire des k plus proches voisins.
- On regarde la classe des voisins les plus proches de l'exemple à tester en se limitant au k plus proche.
- On attribue à l'exemple la classe la plus présente parmi ces voisins par un vote majoritaire.



Hyperparamètres de l'algorithme

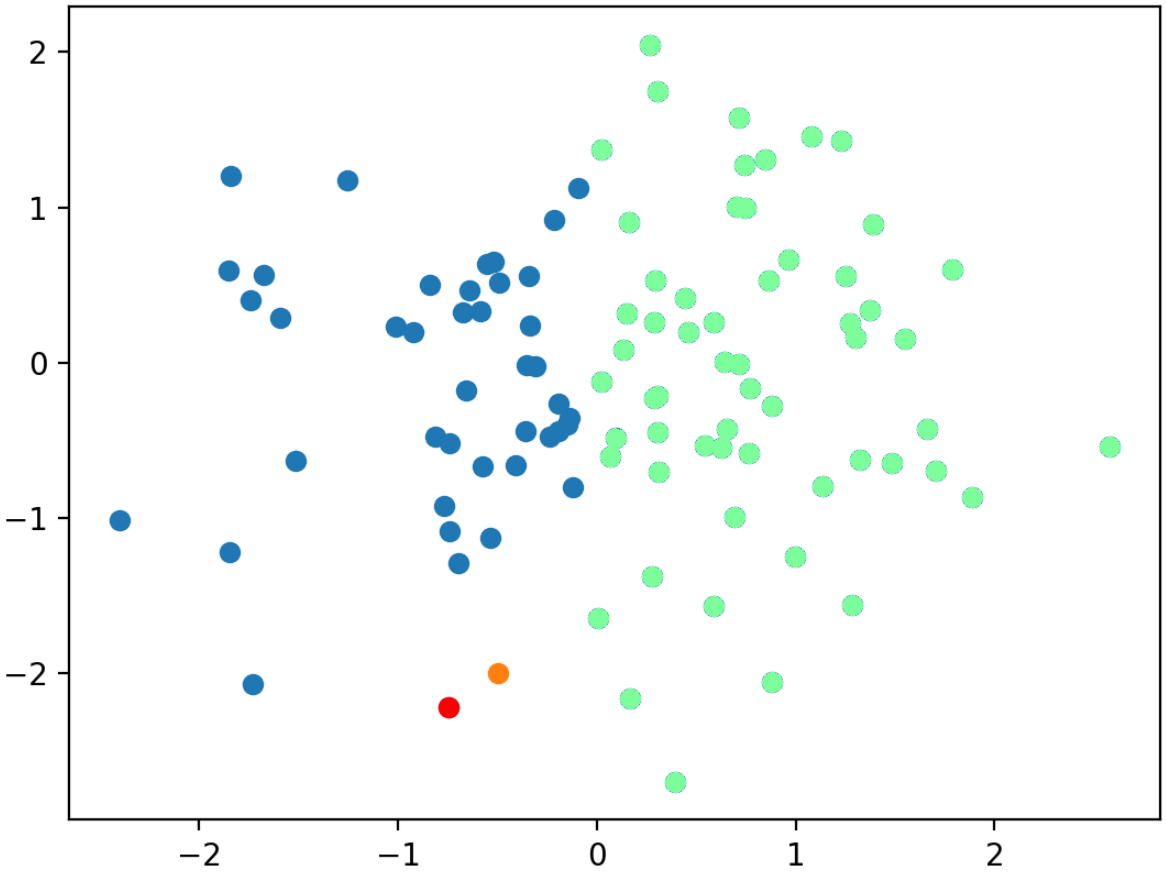
- La valeur de k,
- Le type de distance utilisée (L2,L1,...),
- Règle de pondération (uniforme, distance,...).

Exemple de kNN



Position initiale.

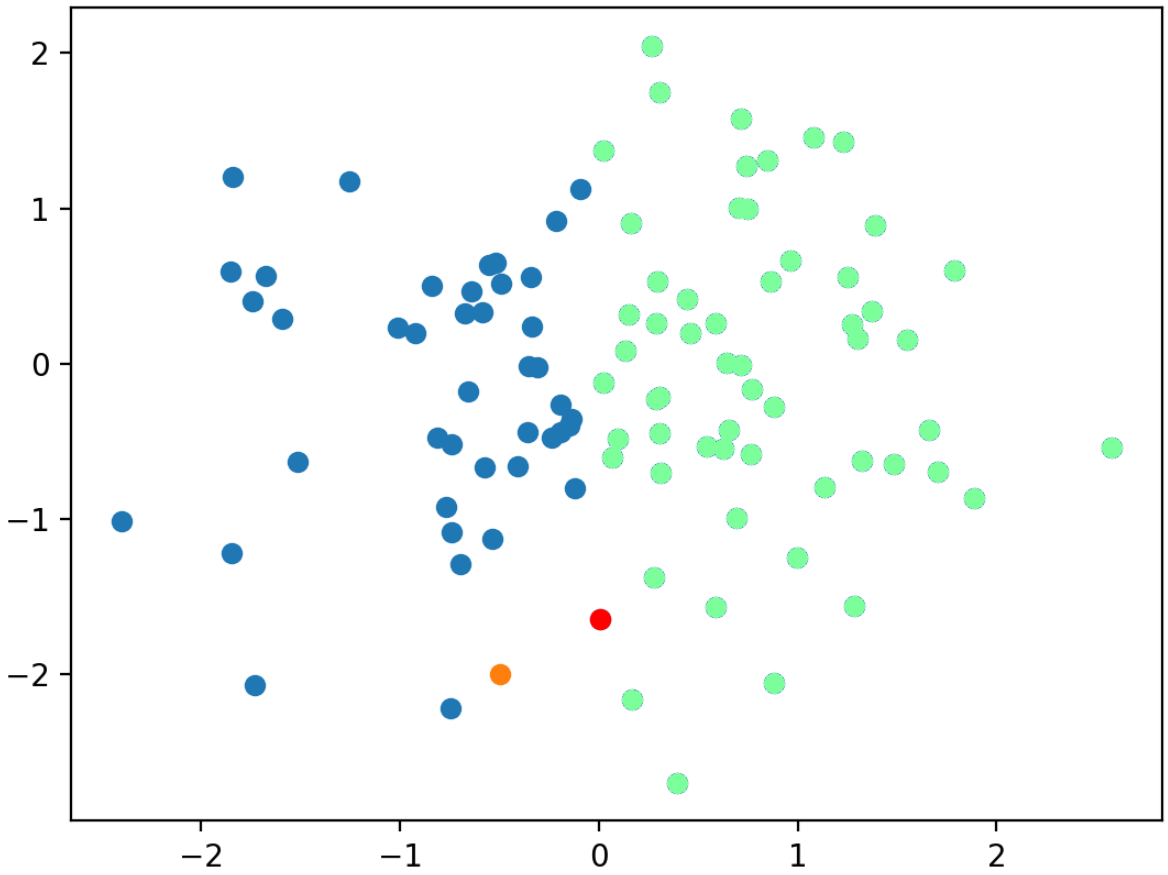
Exemple de kNN



Distance 1er-plus proche voisin : 0,11

Bleu : 1 Vert 0

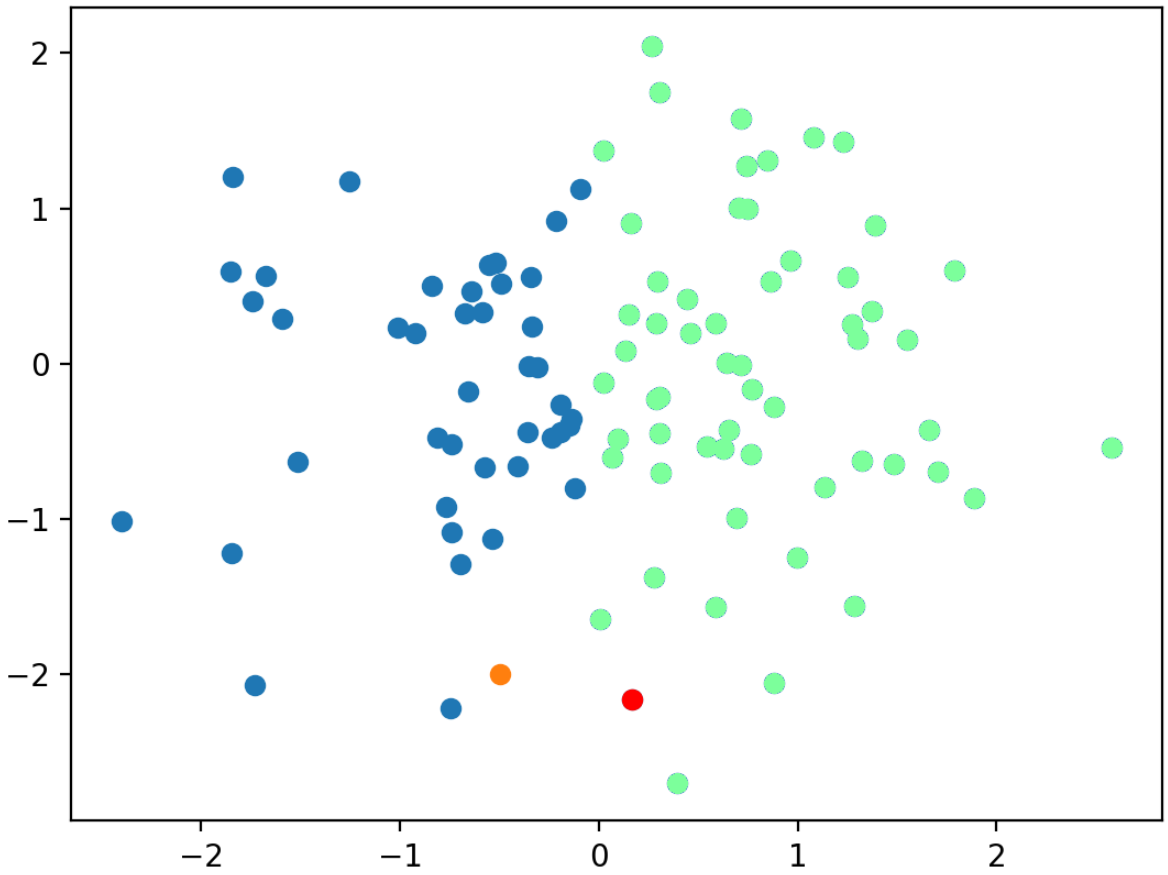
Exemple de kNN



Distance $2^{\text{ème}}$-plus proche voisin : 0,38

Bleu : 1 Vert 1

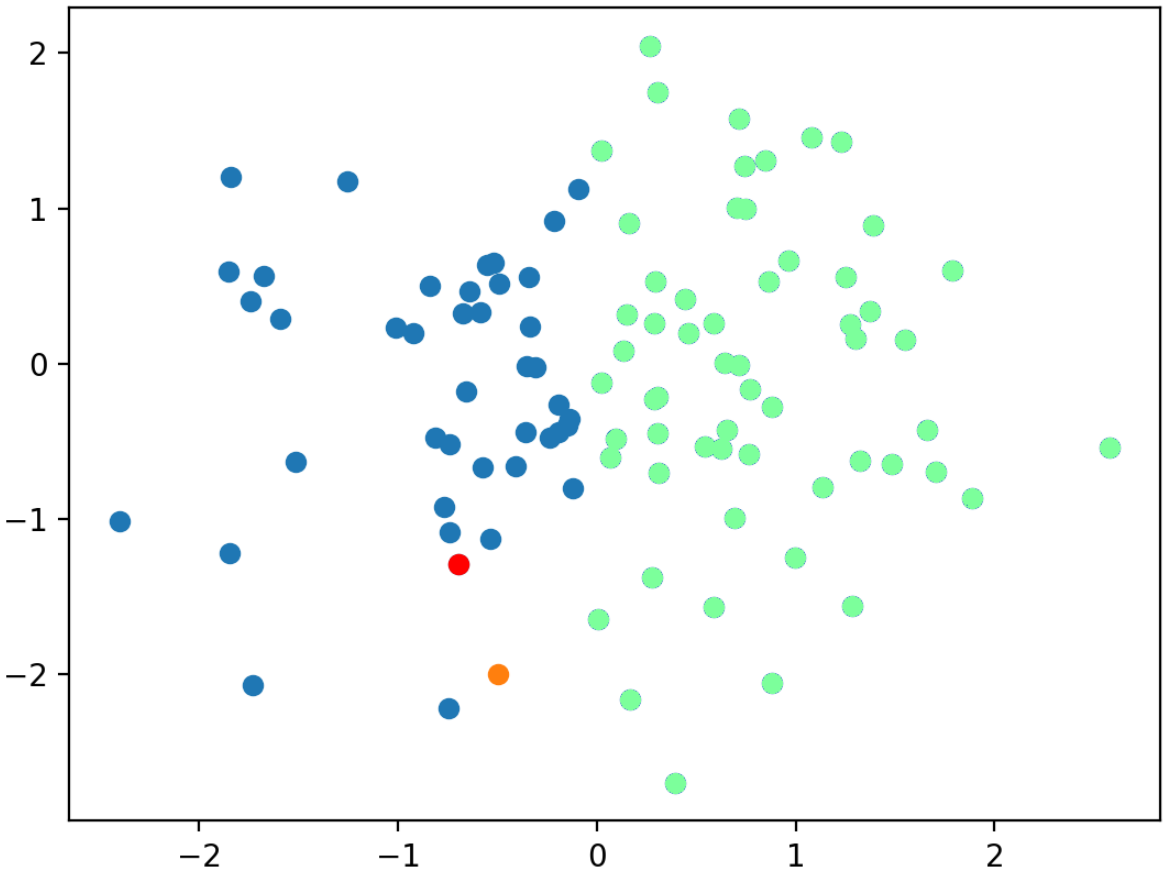
Exemple de kNN



Distance $3^{\text{ème}}$-plus proche voisin : 0,47

Bleu : 1 **Vert 2**

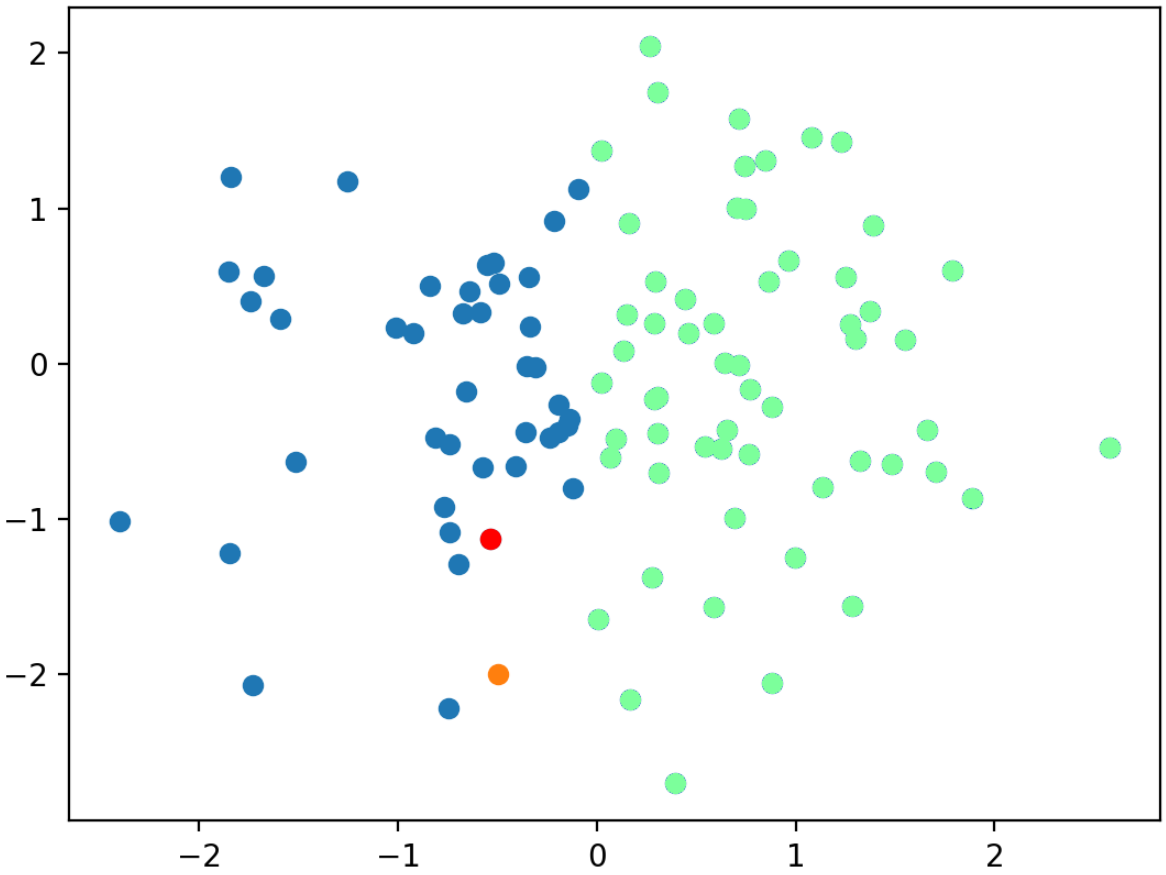
Exemple de kNN



Distance $4^{\text{ème}}$-plus proche voisin : 0,54

Bleu : 2 Vert 2

Exemple de kNN



Distance $5^{\text{ème}}$-plus proche voisin : 0,76

Bleu : 3 Vert 2

Avantages et Inconvénients des kNN

Avantages

- Facile à mettre en place.
- Est assimilable à un classifieur bayésien, optimal au sens de la probabilité d'erreur.

Inconvénients

- Nécessite de garder en mémoire les exemples d'apprentissage,
- Couteux en temps d'exécution si la base d'apprentissage est grande,
- La base d'apprentissage doit être densément répartie dans l'espace de représentation,
- En pratique, difficile à mettre en place pour obtenir de bonne performance.

Sommaire

Une approche bio-inspirée

But

Le cerveau humain excelle dans l'apprentissage de nouvelle situation. Pourquoi ne pas s'en inspirer pour élaborer une méthode d'apprentissage ?

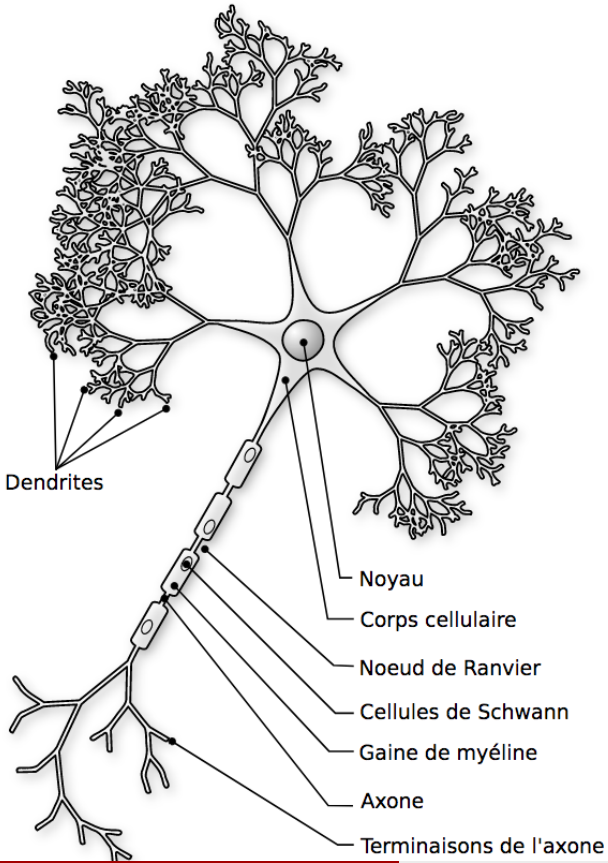
Les avantages d'un cerveau naturel

- Flexible, peut s'adapter à de nouvelles données .
- Robuste et tolérant aux erreurs dans la base d'apprentissage.
- Peu fonctionner avec des données incomplètes ou en partie fausse.
- A des grandes capacités d'apprentissage.
- Est rapide et utilise un système massivement parallèle.
- Repose sur des éléments dont le fonctionnement est relativement simple et facilement reproductible.

Du neurone réel au neurone formel

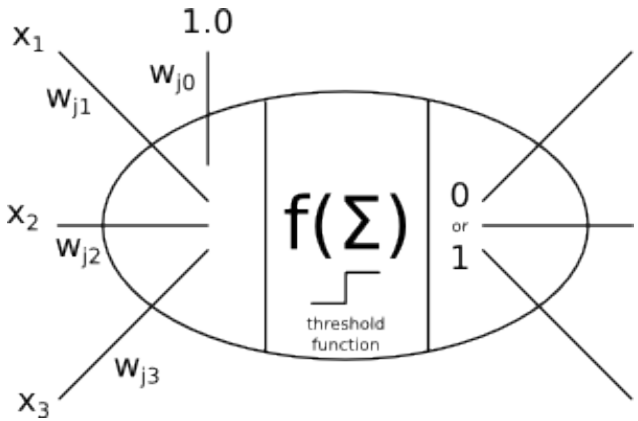
Composition simplifié d'un cerveau humain

- 100 milliards de neurones,
- en moyenne 10.000 connexions par neurone.

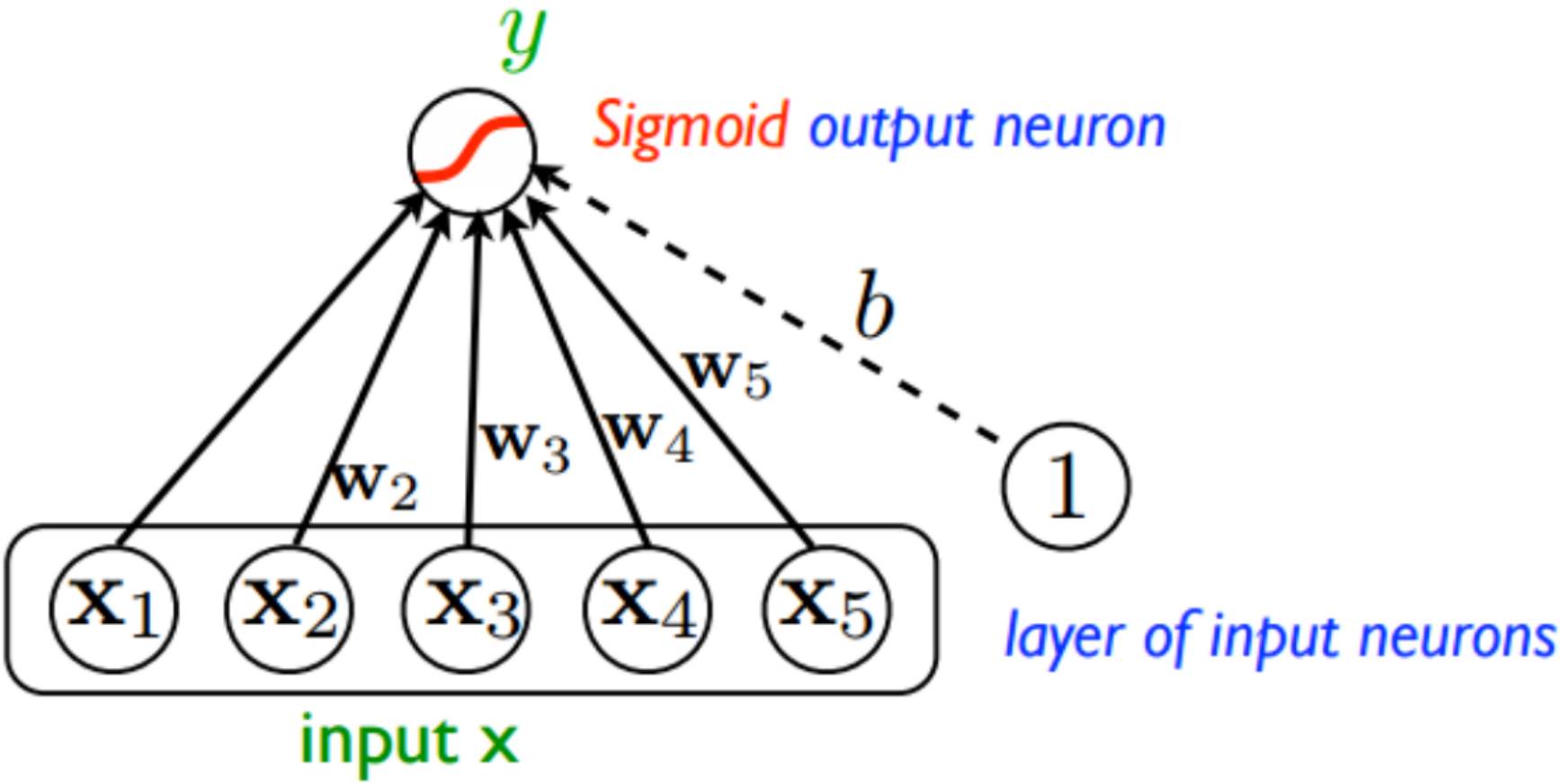


Le neurone formel

- Un neurone possèdent des entrées.
- Le signal de chaque entrée est pondéré par un poids.
- La sortie dépend des entrées et des poids.
- La sortie est un signal d'activation ou non du neurone.



Le neurone formel



Neurone formel est une classification linéaire

Les entrées

Chaque exemple de la base d'apprentissage correspond à un vecteur $x \in \mathrm{R}^d$ d'entrée du neurone. Pour chaque exemple d'apprentissage on peut donc associer un point dans un espace à d dimension.

Rôle du neurone

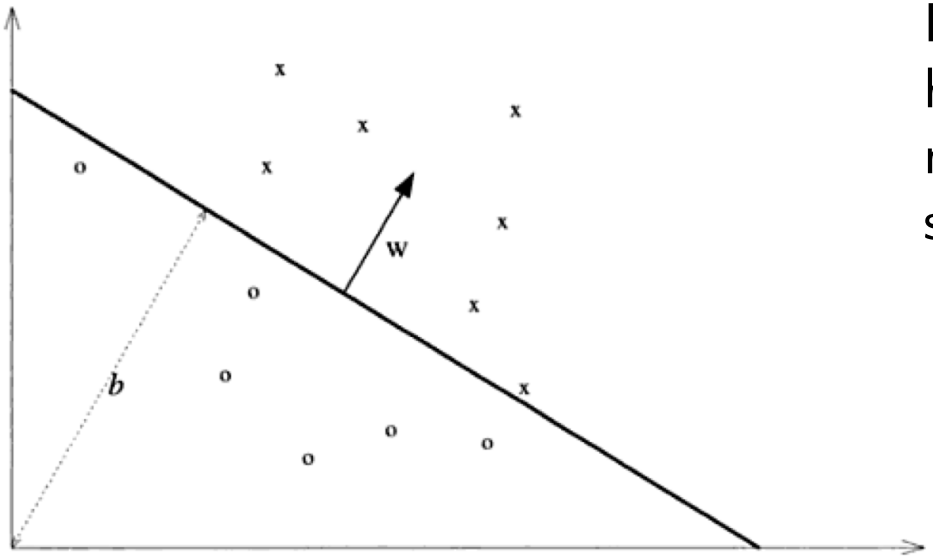
L'objectif du neurone est de trouver la séparation entre deux classes. Les points activant le neurone et les points ne devant pas l'activer. Il résout un problème de classification binaire.

Type de frontière décrite par un neurone

Le signal post-synaptique correspond au résultat d'une fonction de type $b + \langle w, x \rangle$. Cela correspond à une frontière linéaire, définis par le vecteur orthogonal w et le biais b.

Un neurone formel : la résolution d'une classification linéaire à deux classes

Objectif



L'objectif est de trouver un hyperplan qui sépare au mieux nos deux classes. Le classifieur sera alors la fonction :

$$f(x) = \begin{cases} x \text{ si } x \text{ est au dessus} \\ \quad \text{de l'hyperplan} \\ o \text{ si } x \text{ est en dessous} \\ \quad \text{de l'hyperplan} \end{cases}$$

Formalisation mathématique

$$f(x) = \begin{cases} 1 \text{ si } \langle w, x \rangle + b > 0 \\ -1 \text{ sinon} \end{cases}$$

Le Perceptron de Rosenblatt (1956)

Historique

Le perceptron est un des premiers et des plus simples réseaux de neurones. Il fut inventé par Frank Rosenblatt en 1956. C'est une solution pour calculer un classifieur linéaire pour un ensemble de données d'apprentissage à deux classes. Cette méthode est inspirée du fonctionnement des neurones réels.

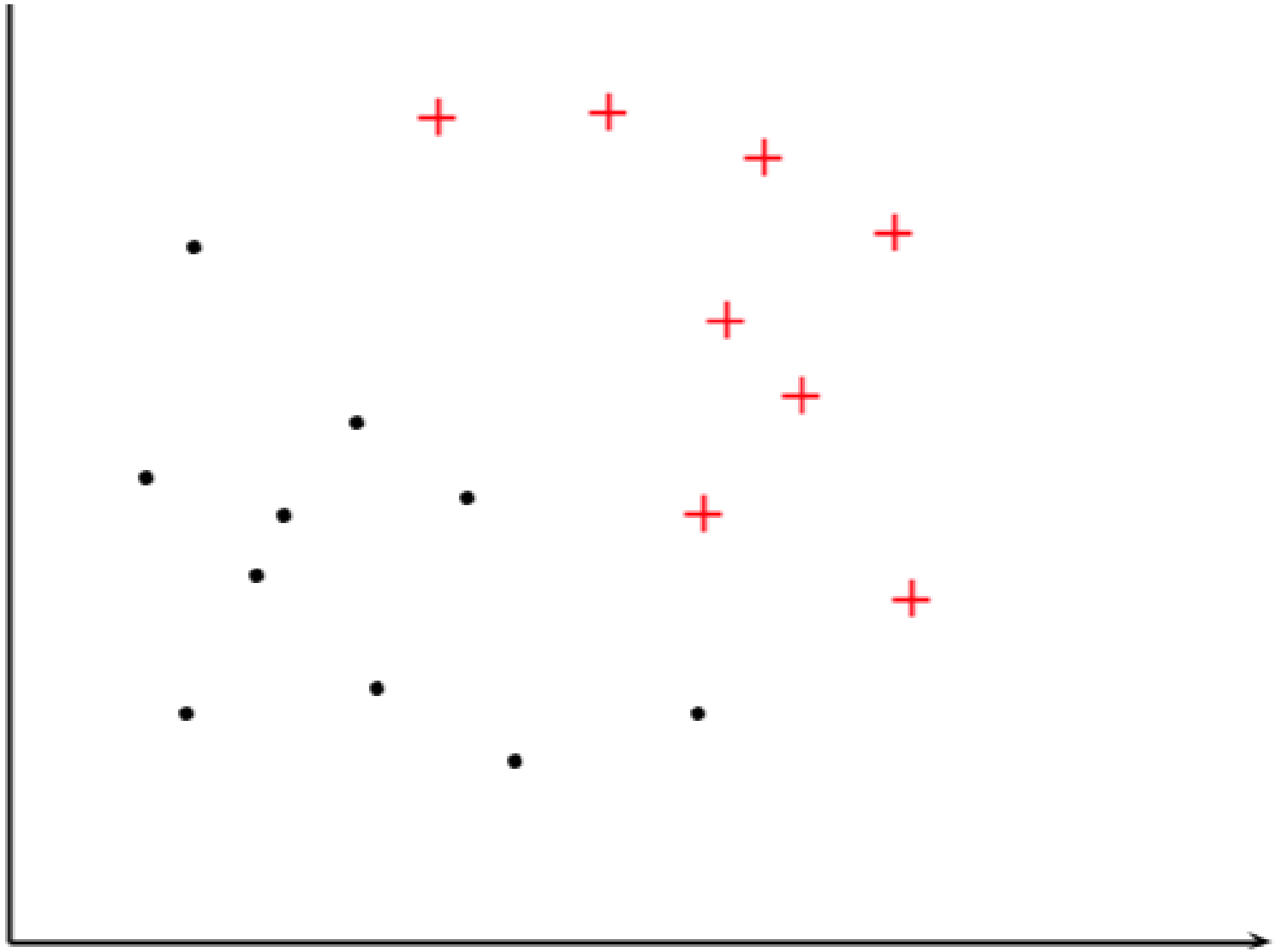
Principe

- Algorithme itératif
- Pour chaque exemple mal classé, on corrige l'hyperplan pour faire passer l'exemple de l'autre côté de l'hyperplan.

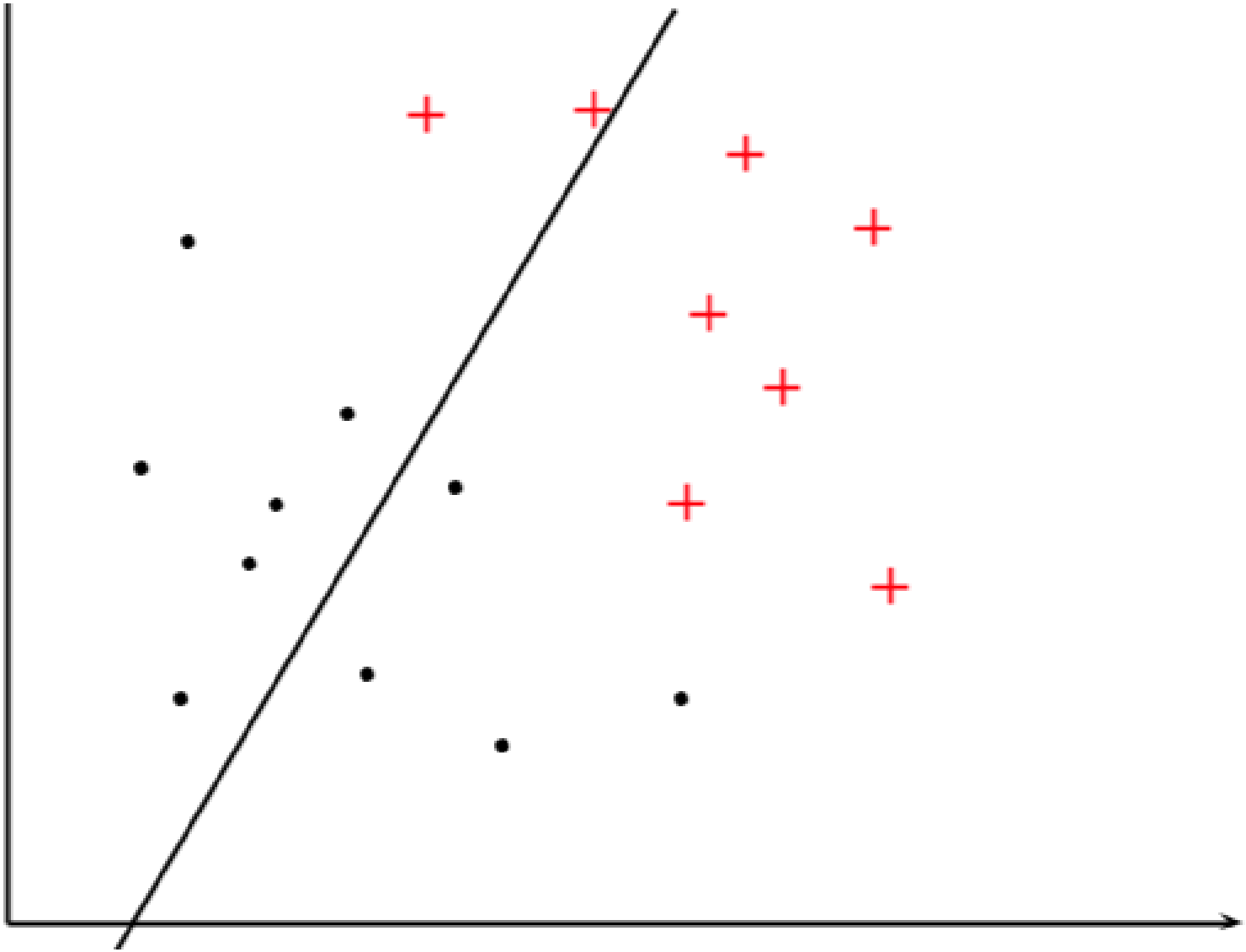
Pour approfondir le sujet

The perceptron : a probabilistic model for information storage and organization in the brain. F.Rosenblatt, Psychological Review, Vol. 65, n 6, 1958.

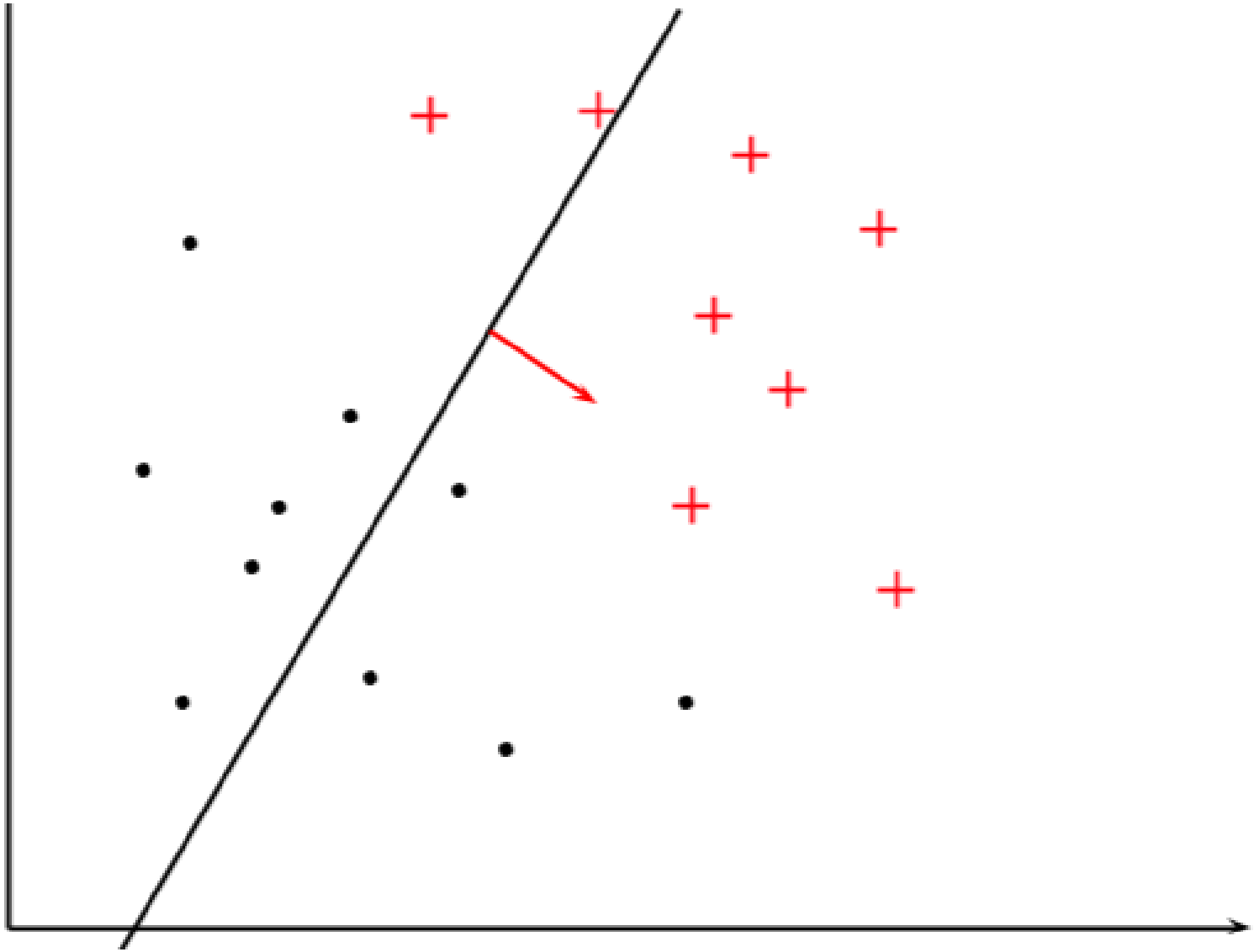
Exemple



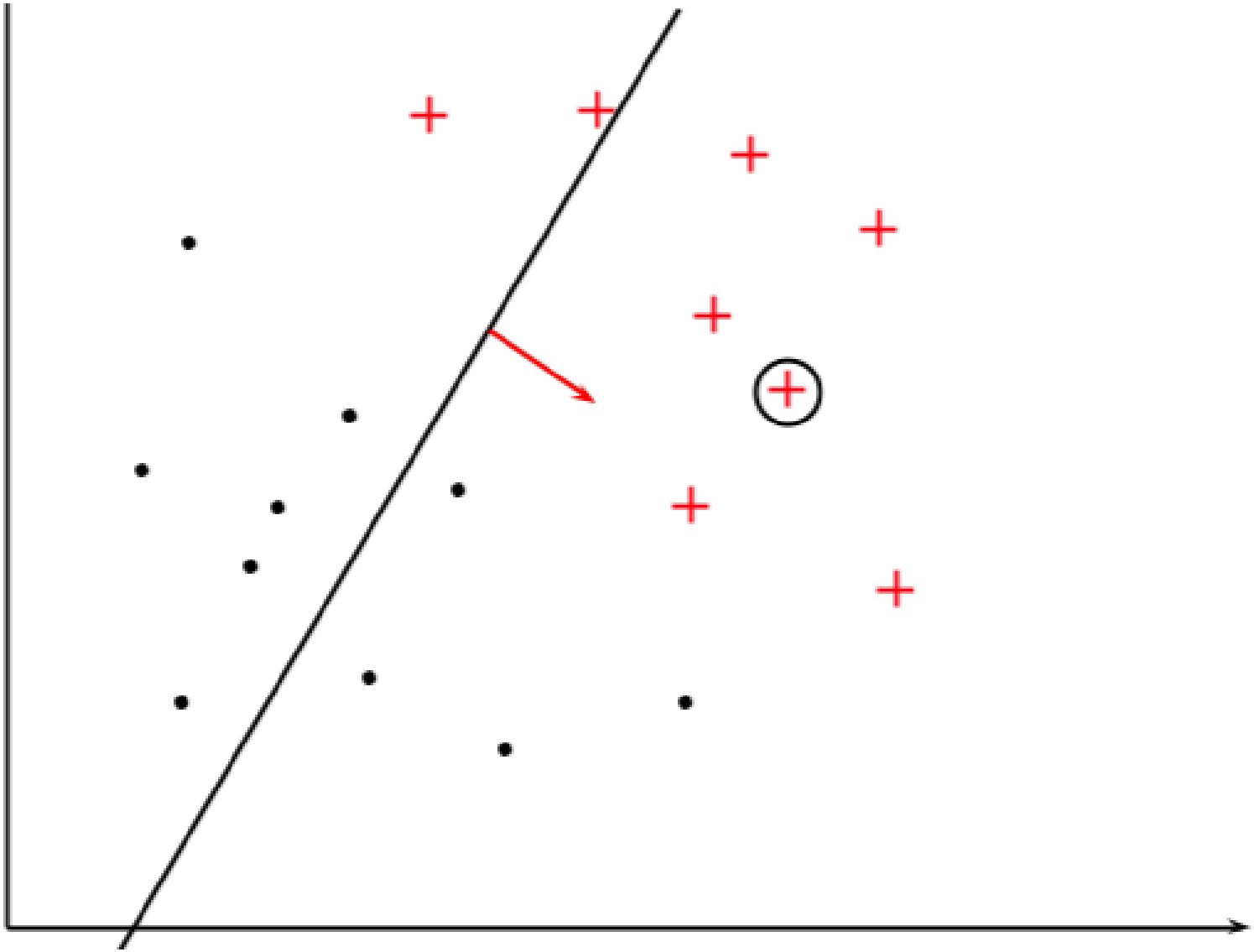
Initialisation du perceptron par un hyperplan aléatoire



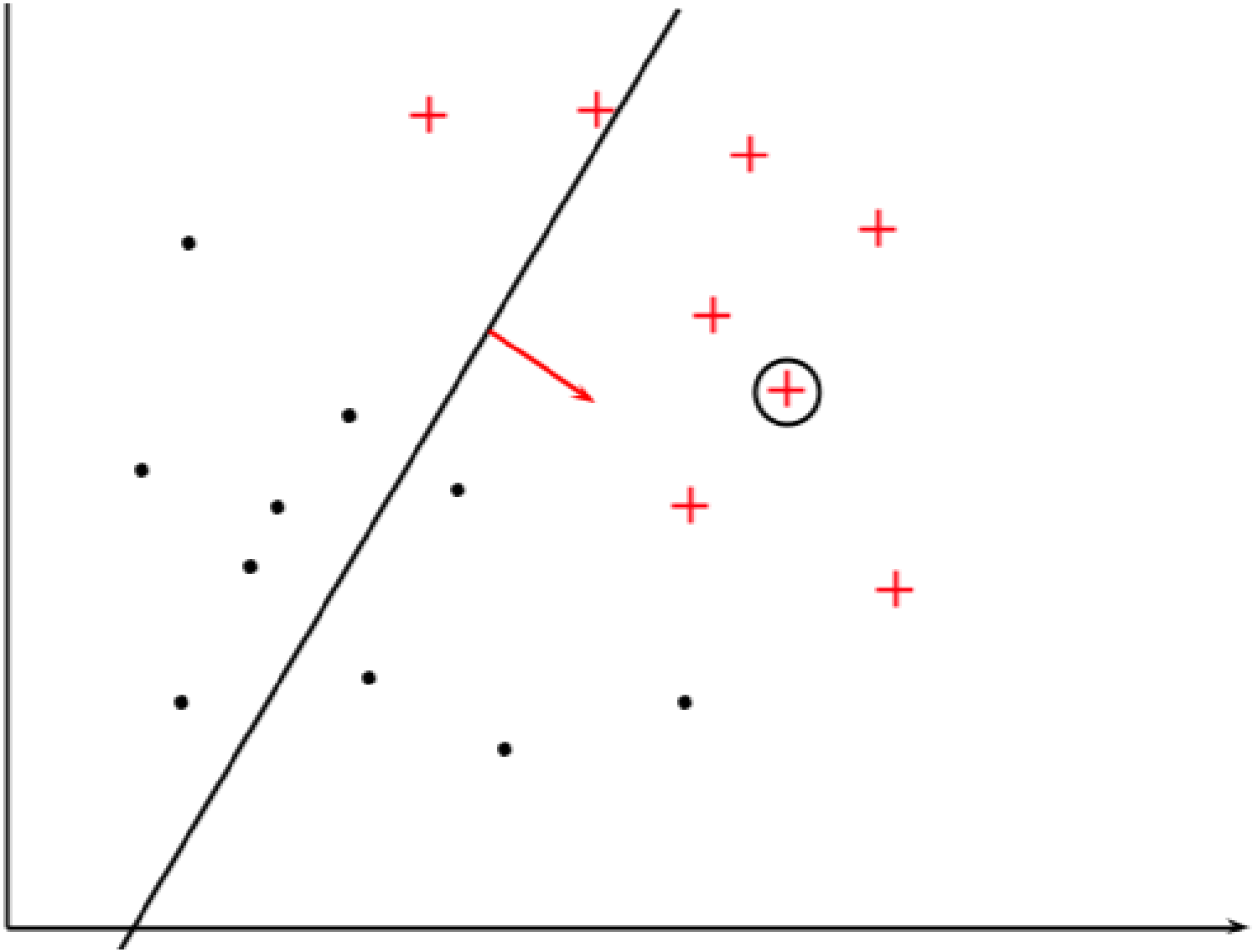
Direction de l'hyperplan



Choix d'un premier exemple

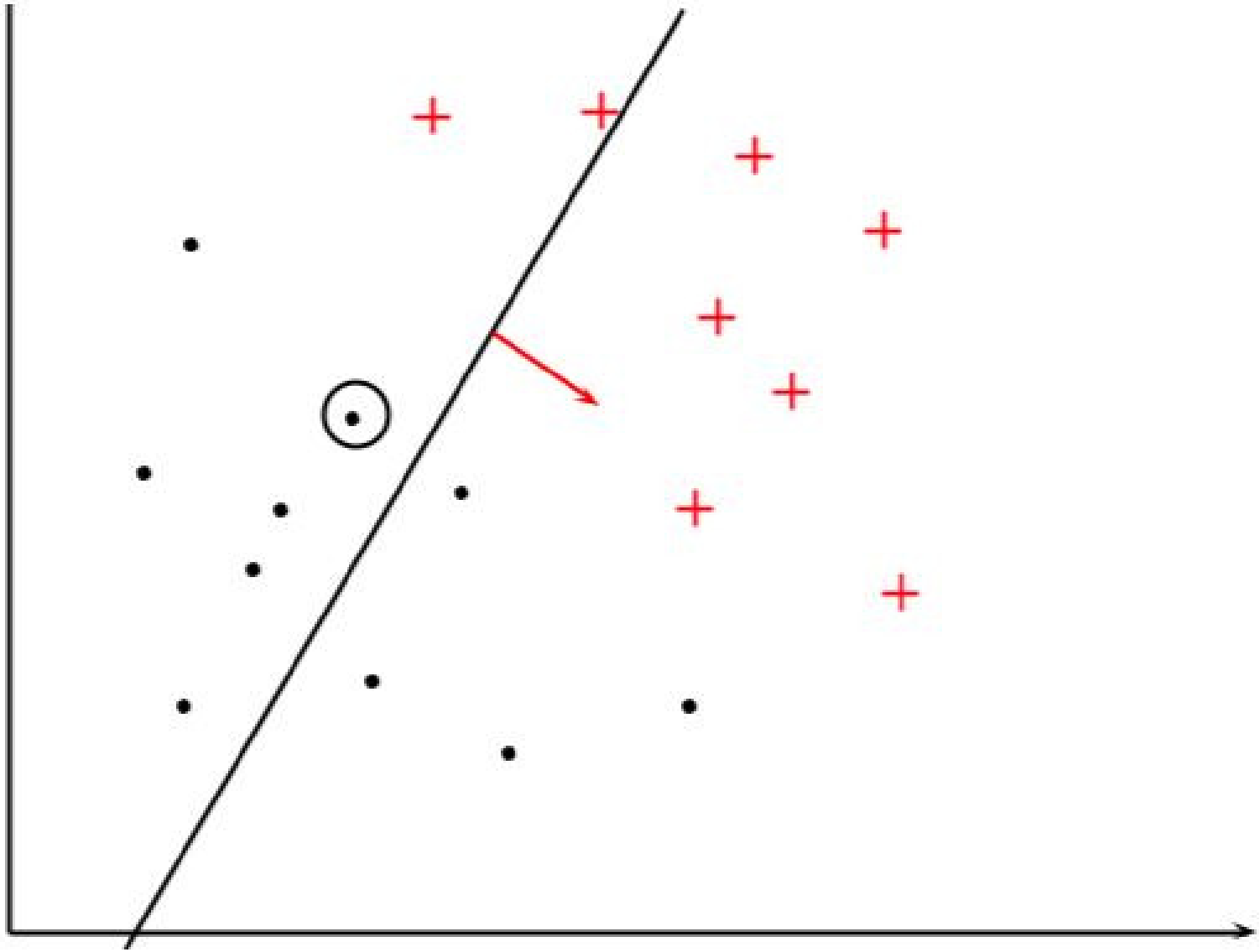


Choix d'un premier exemple

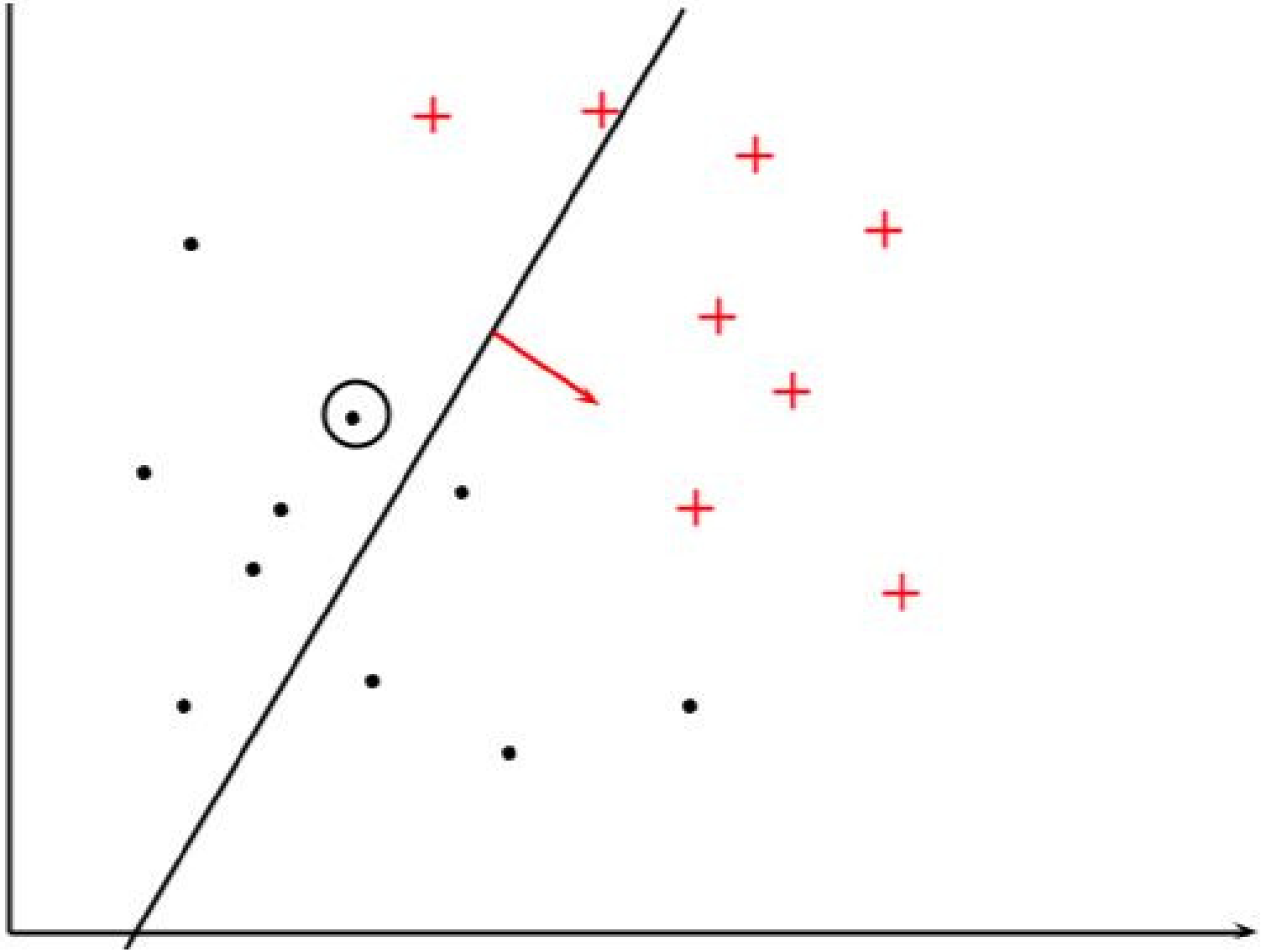


$\Longrightarrow$ L'hyperplan classe correctement cet exemple. Il n'y a rien à faire.

Choix d'un deuxième exemple

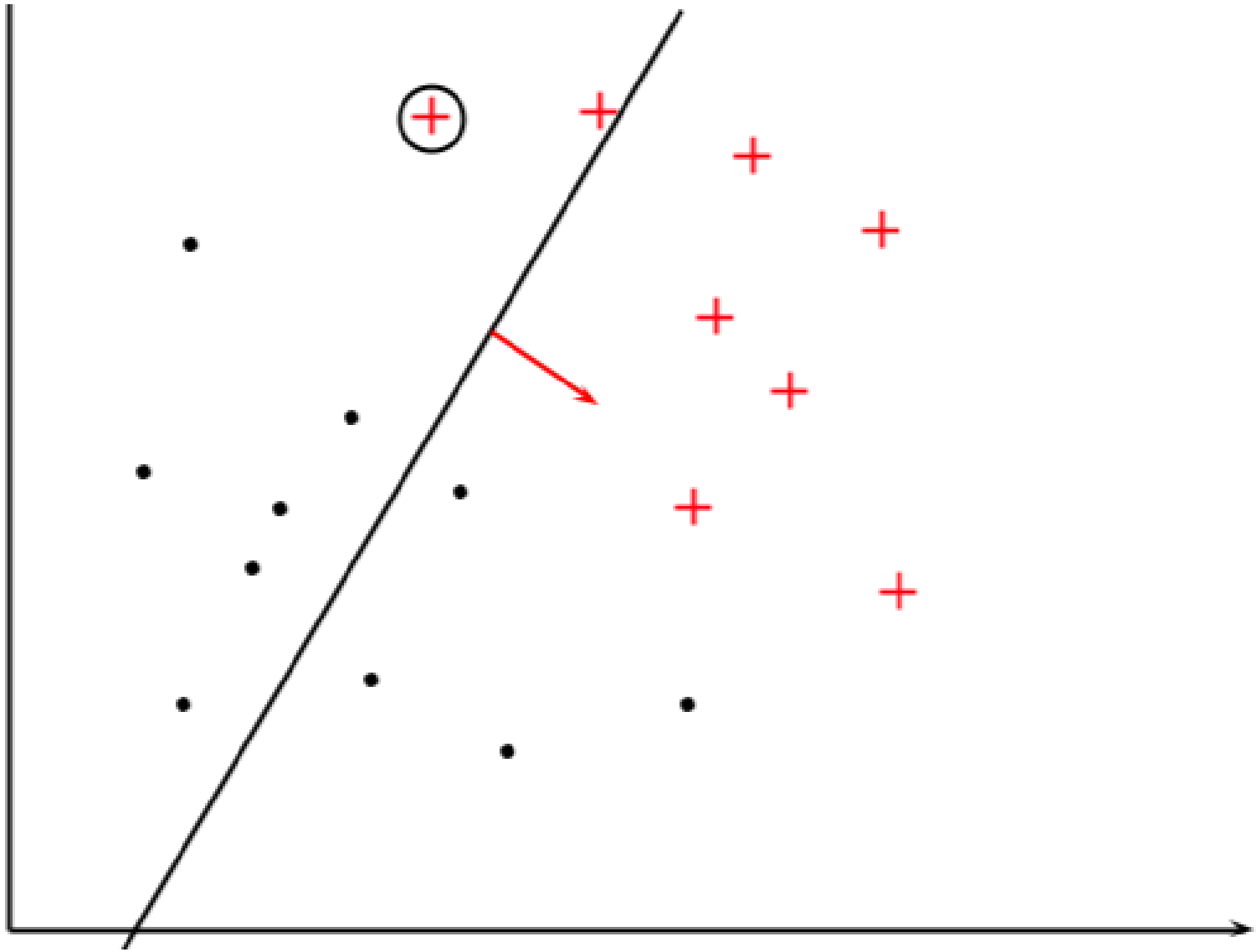


Choix d'un deuxième exemple

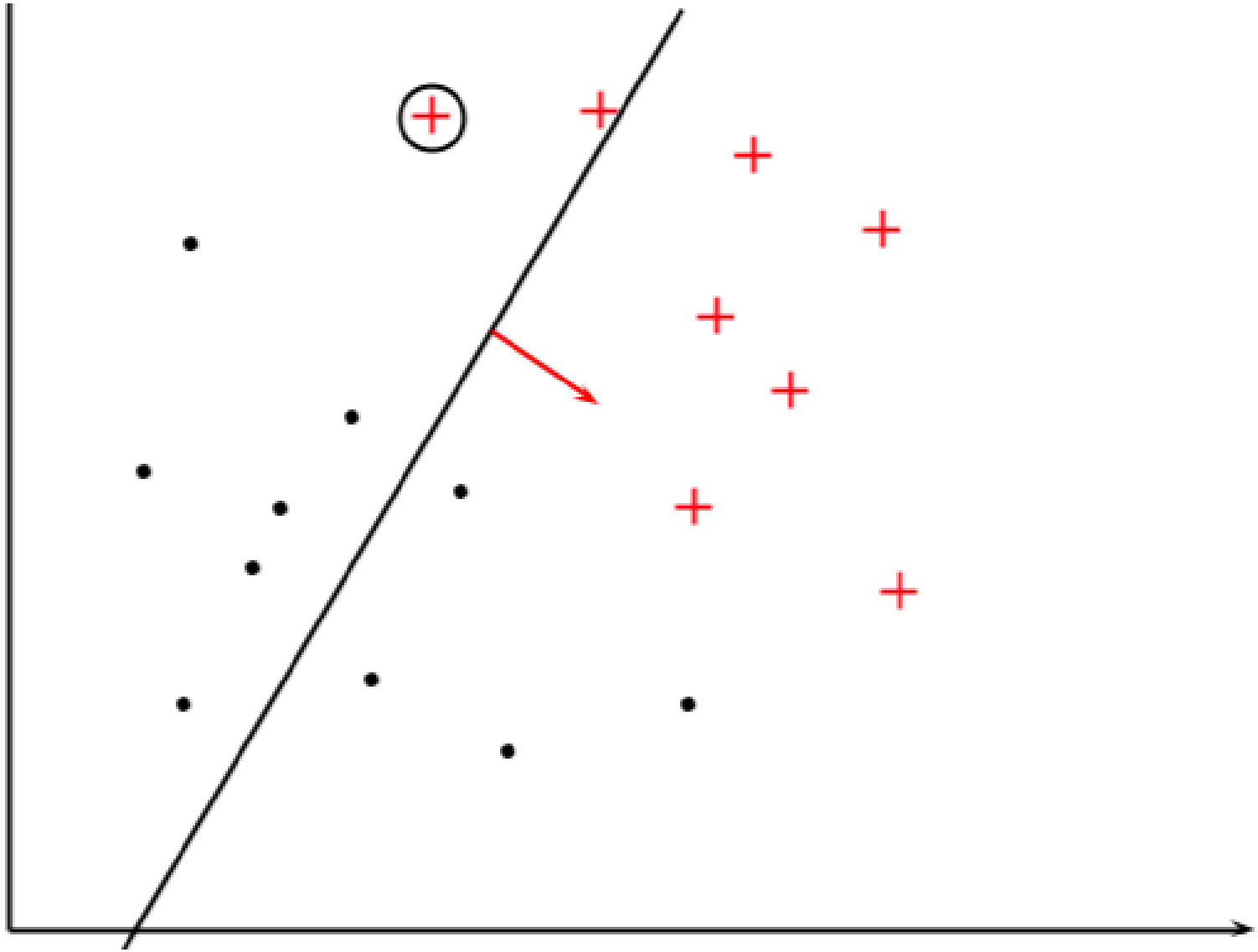


$\Longrightarrow$ L'hyperplan classe correctement cet exemple. Il n'y a rien à faire.

Choix d'un troisième exemple

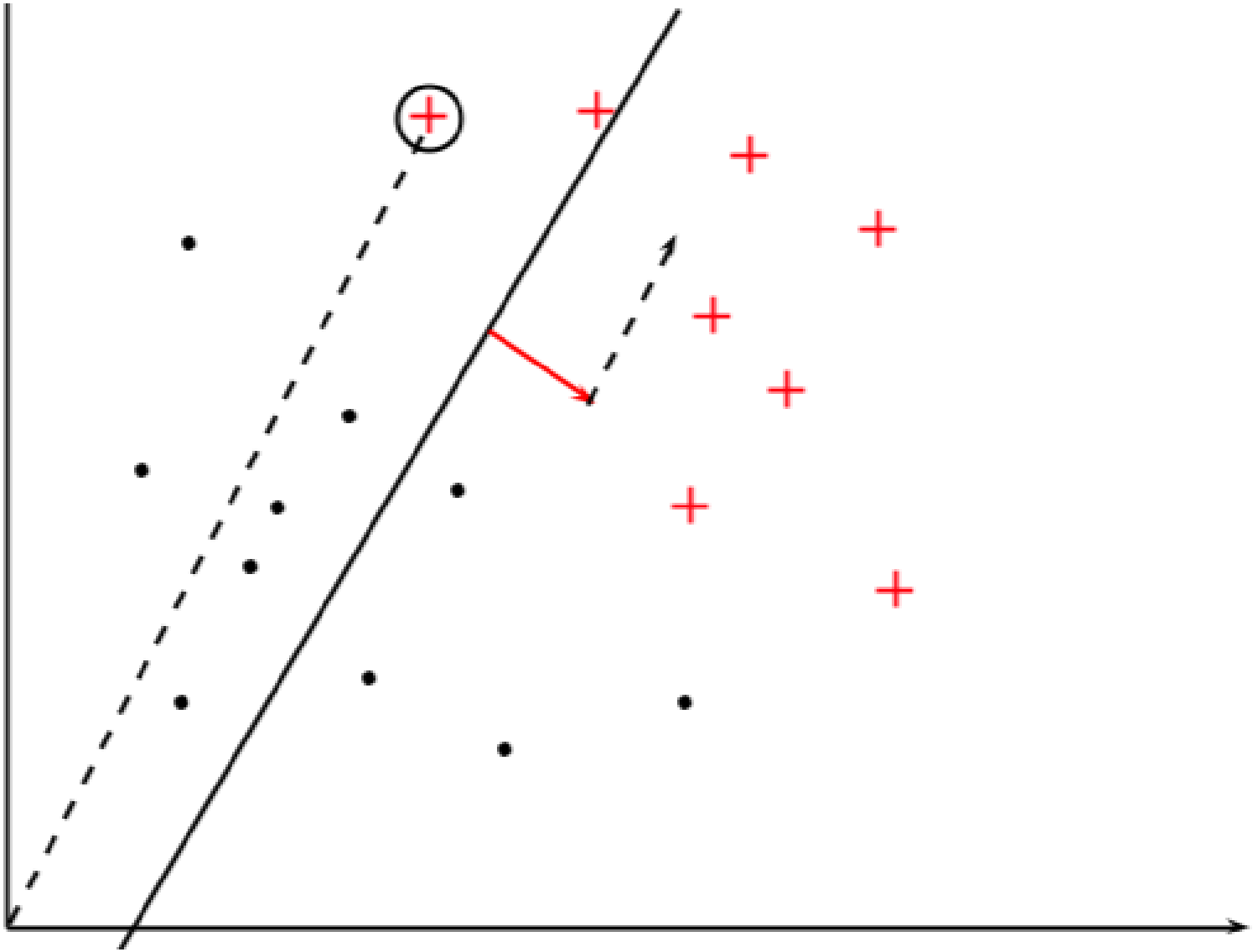


Choix d'un troisième exemple

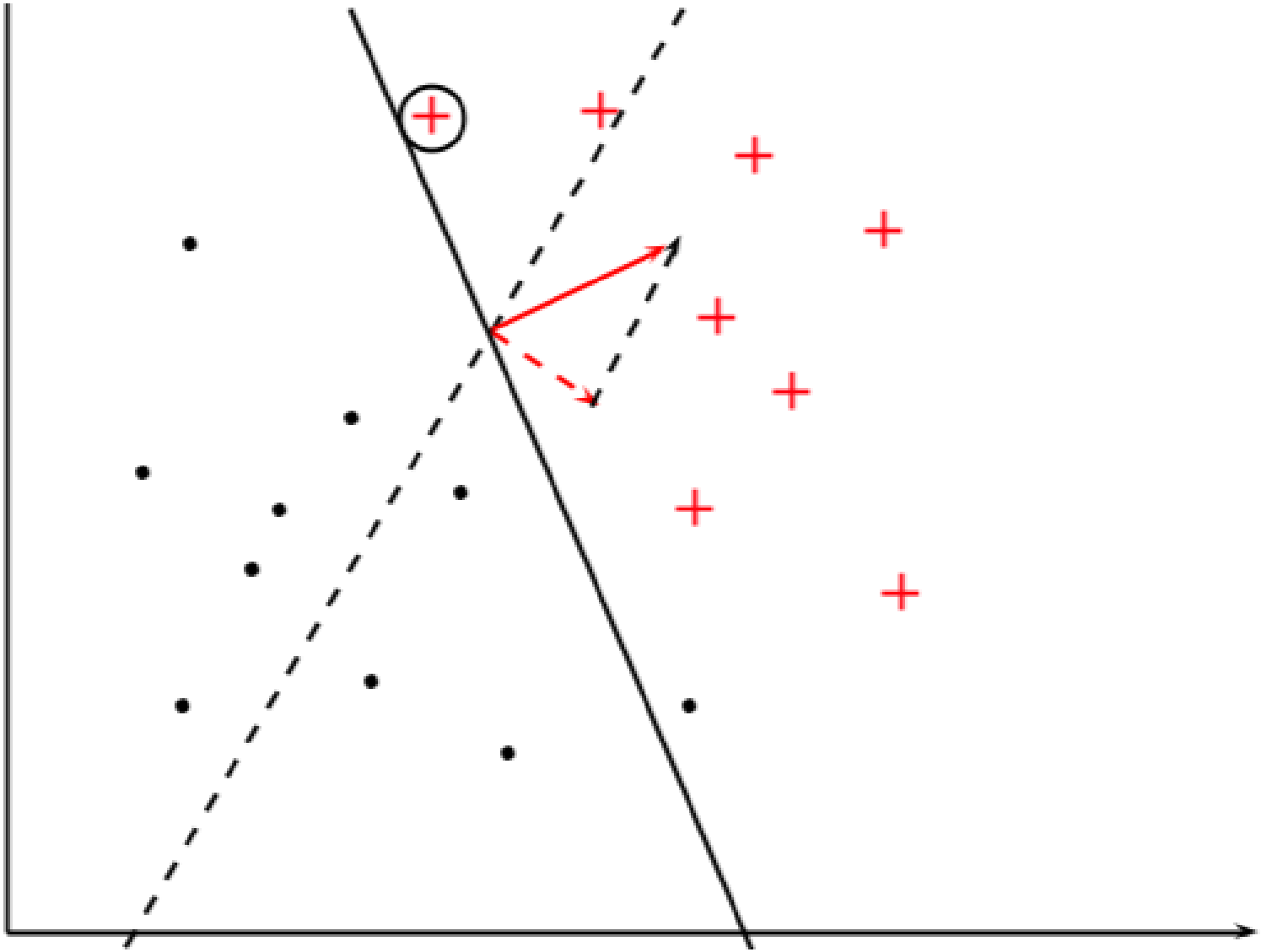


$\Longrightarrow$ L'hyperplan **ne** classe **pas** correctement cet exemple. Il faut le corriger.

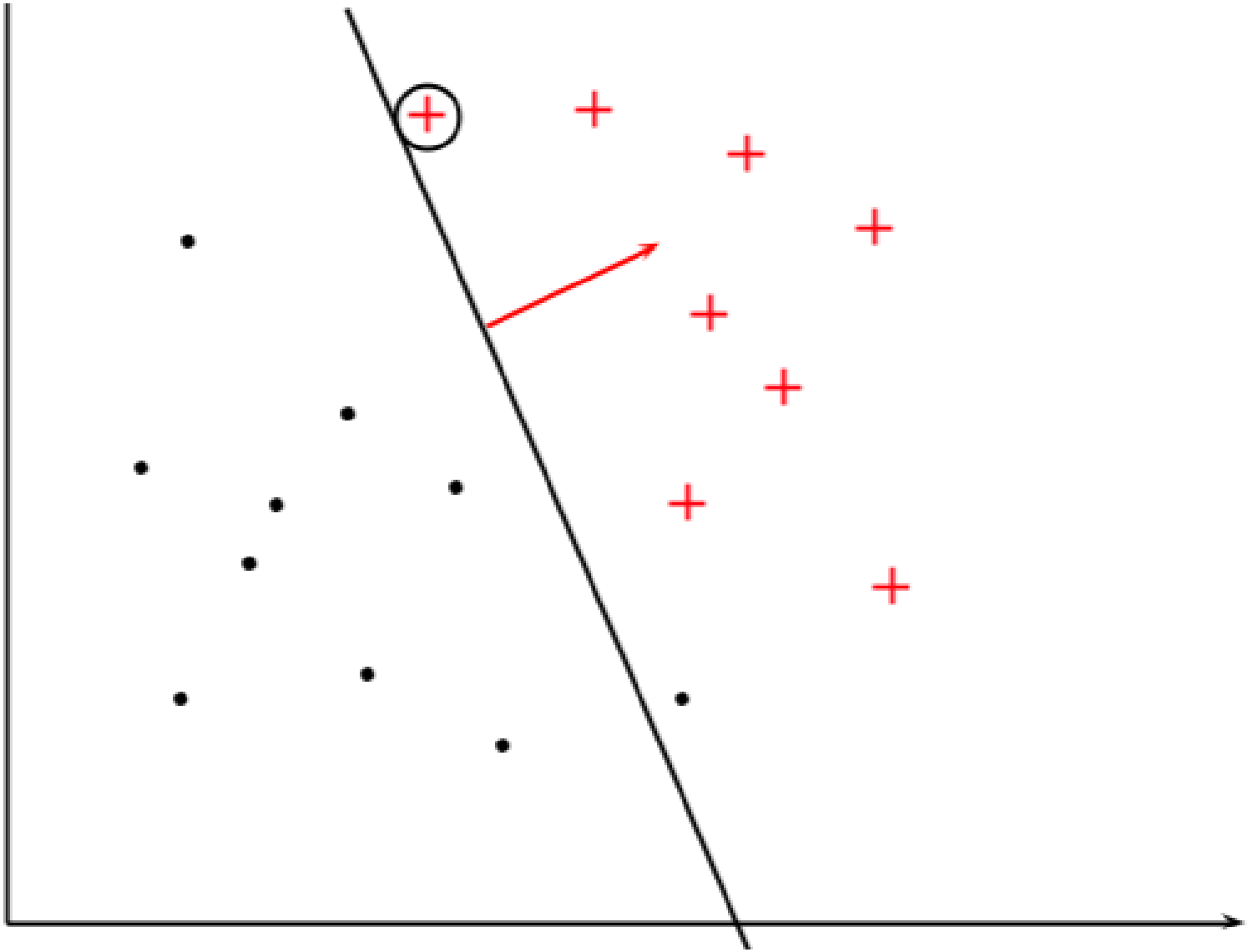
Décalage dans la direction du positif faux



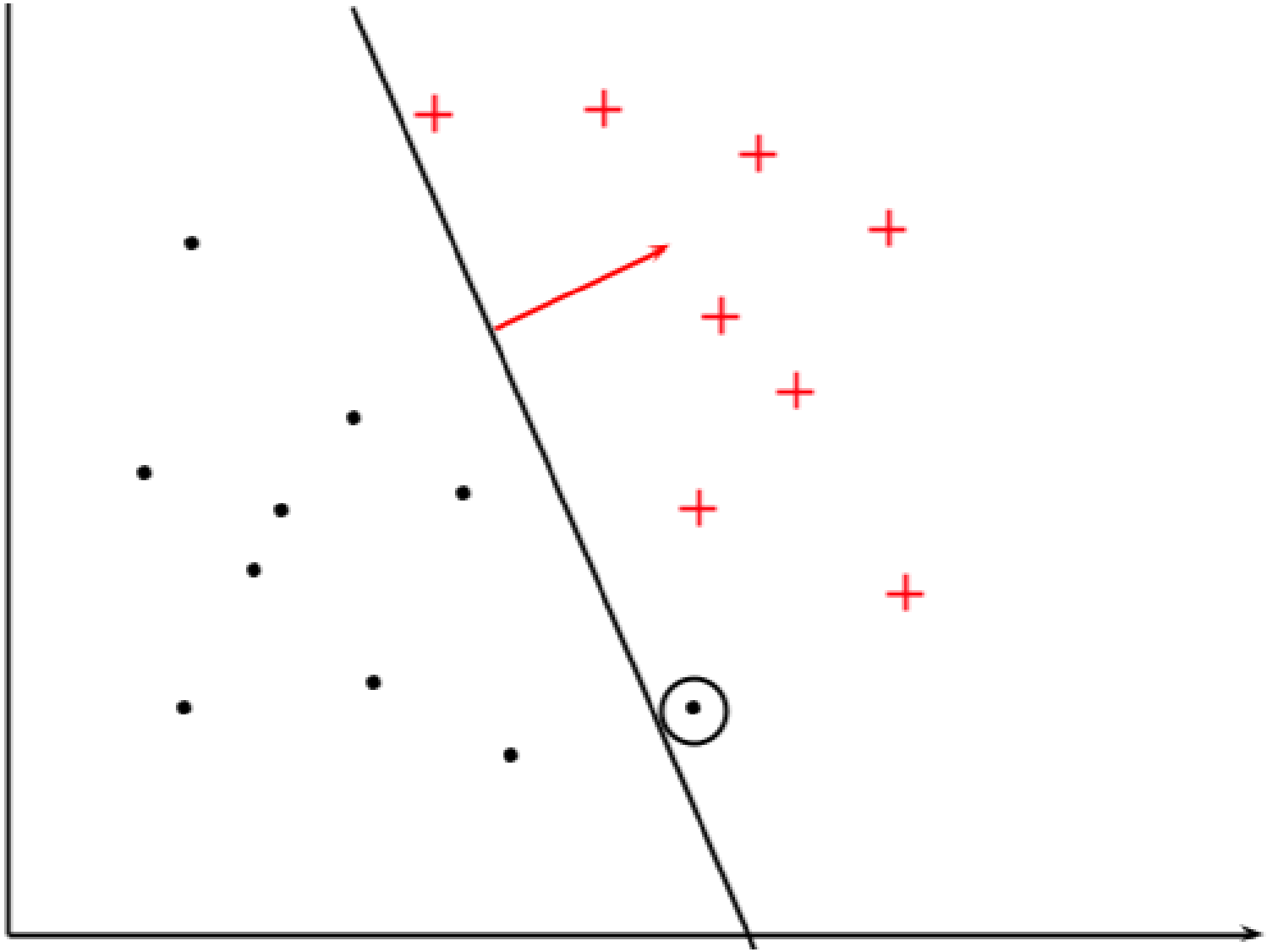
Décalage dans la direction du positif faux



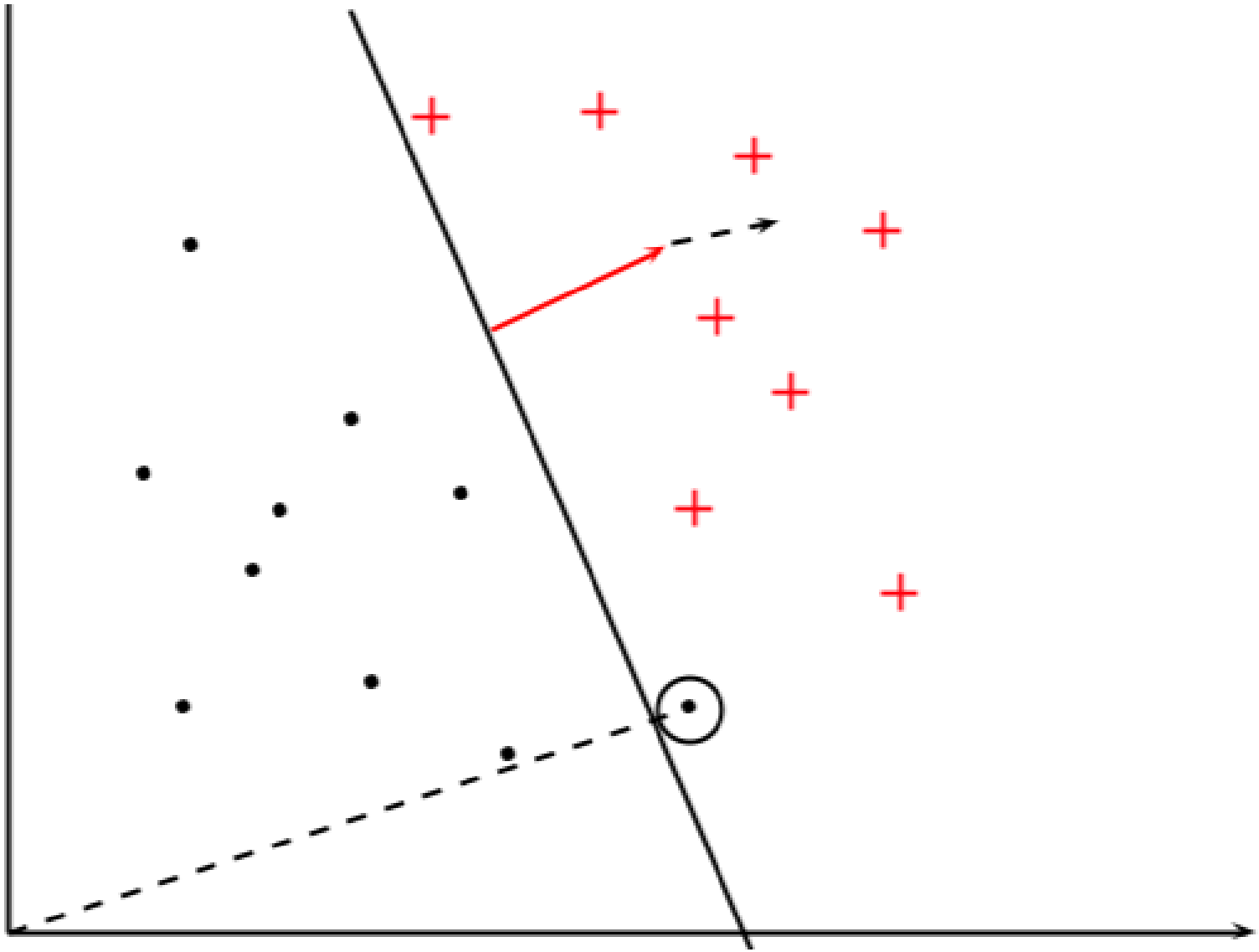
Nouvel hyperplan



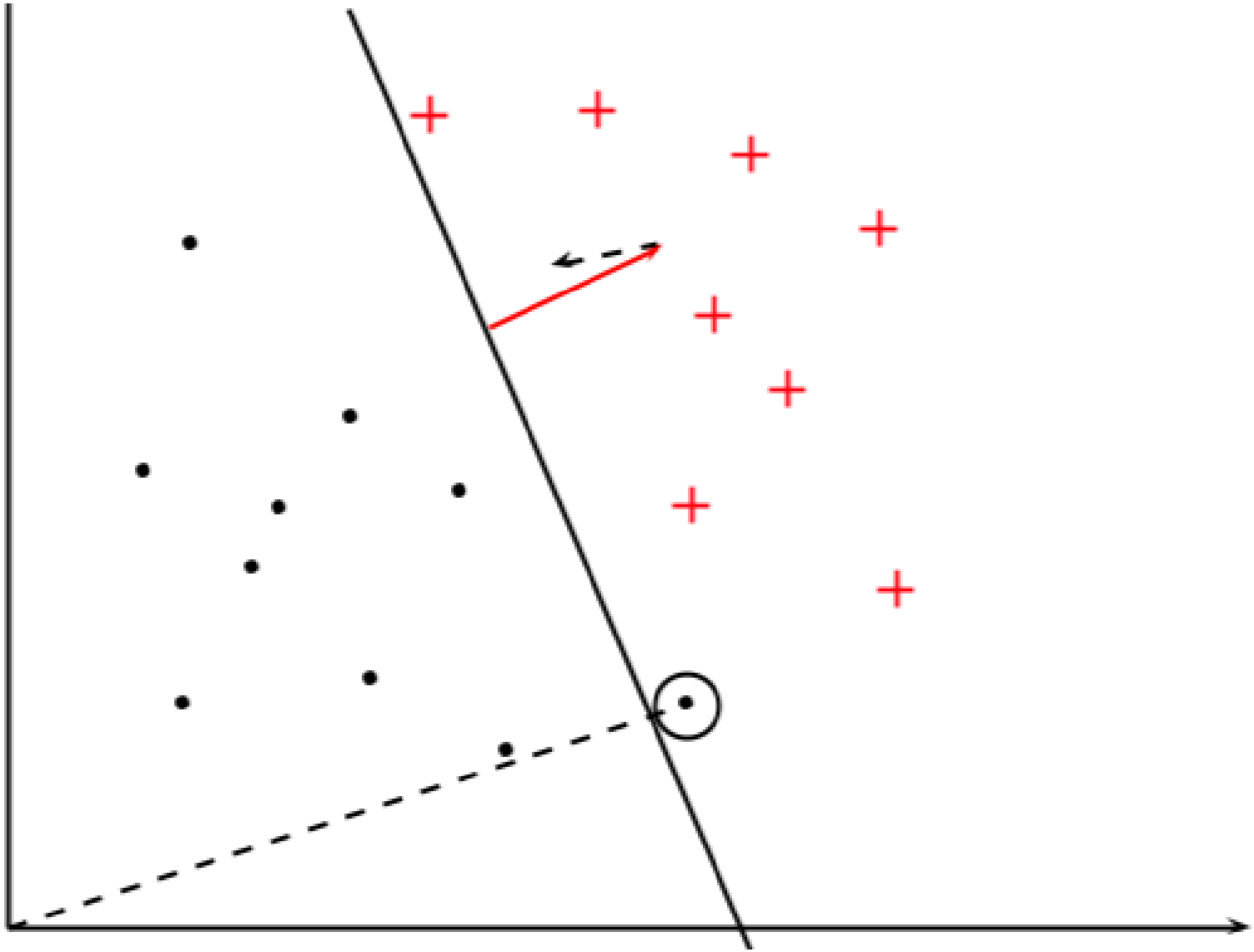
Choix d'un quatrième exemple



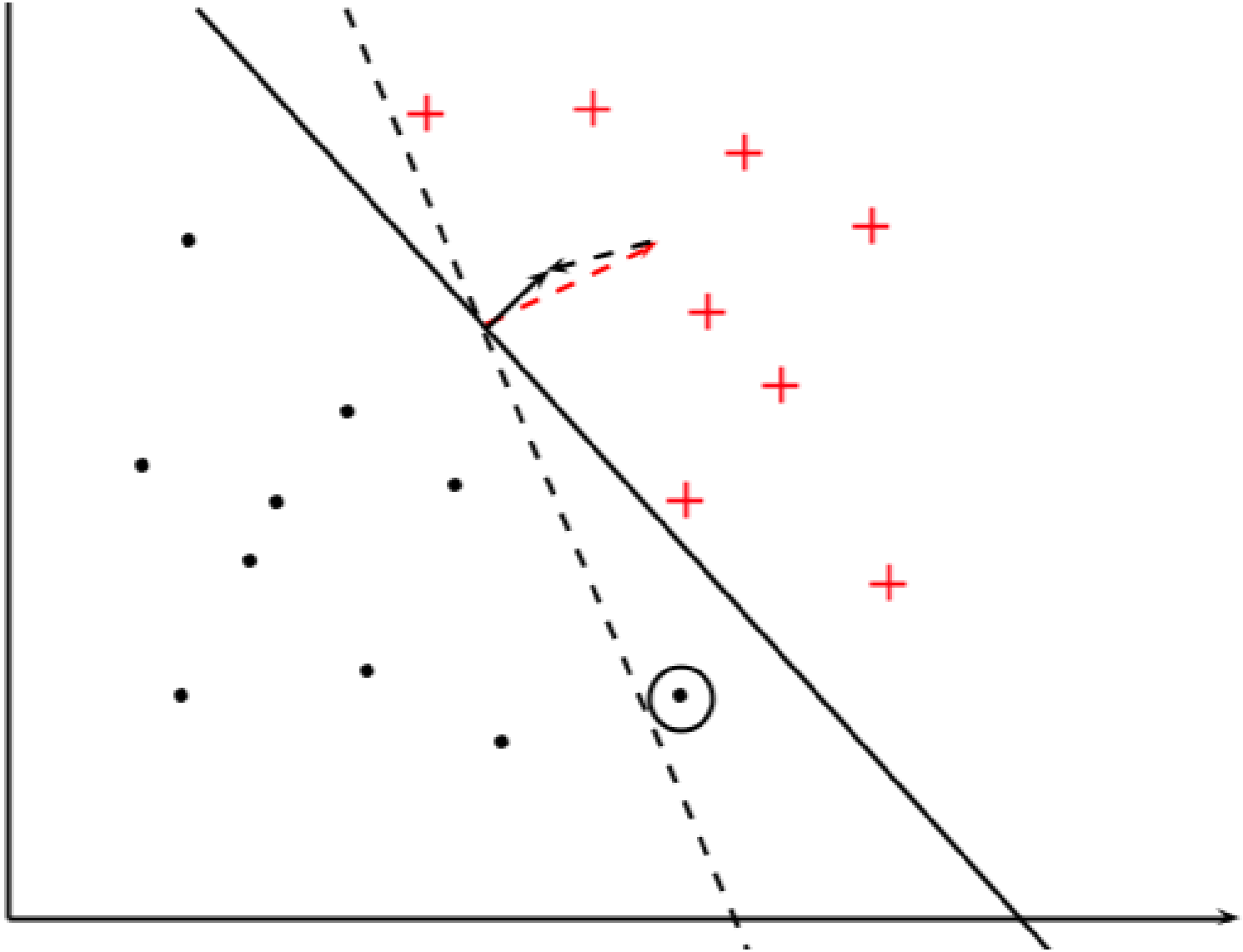
Décalage dans la direction inverse du négatif faux



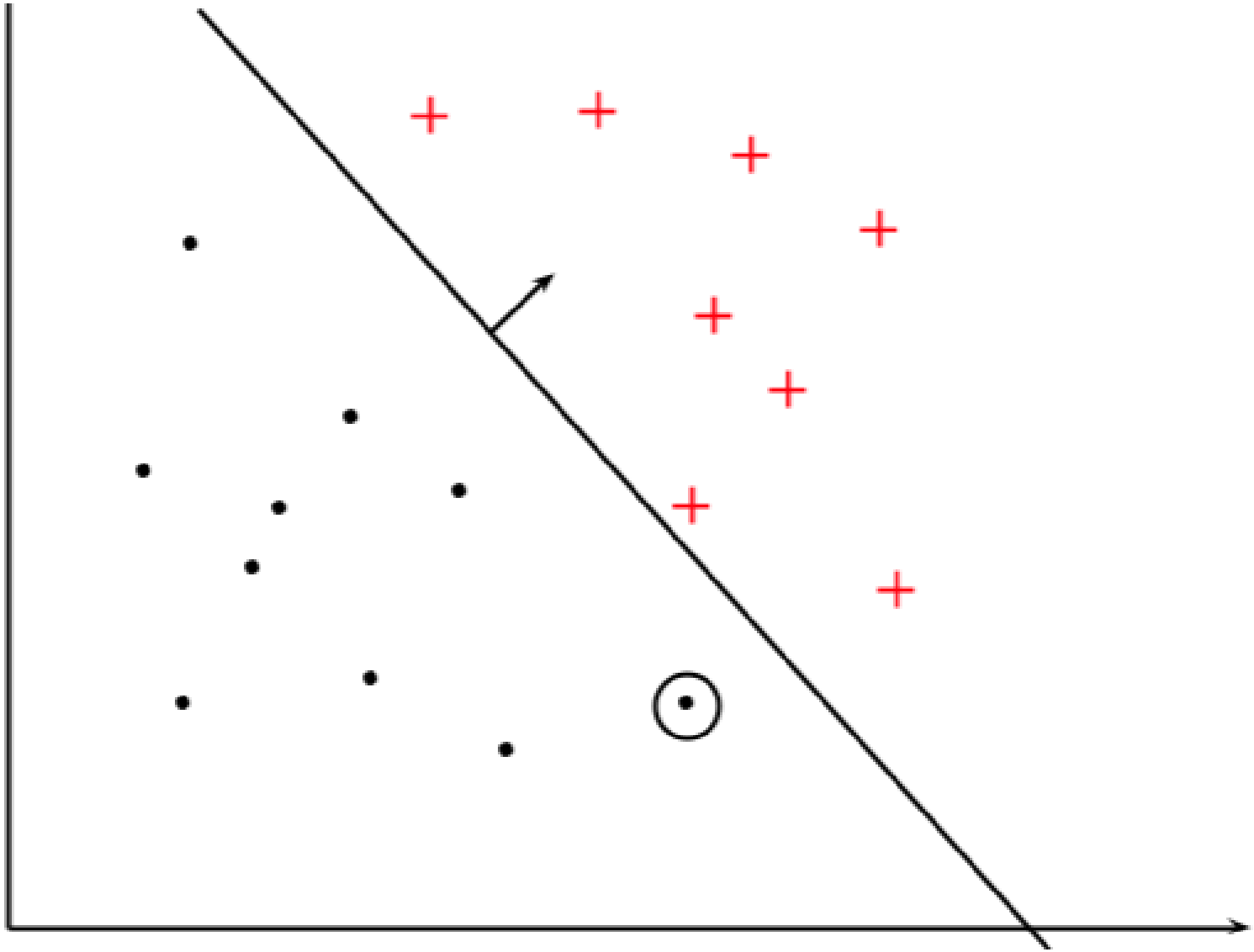
Décalage dans la direction inverse du négatif faux



Décalage dans la direction inverse du négatif faux



Décalage dans la direction inverse du négatif faux



Algorithme du Perceptron (dans le primal)

Algorithme

Data: Un ensemble $\mathcal{S}$ d'apprentissage linéaire séparable et un taux $\eta \in \mathbb{R}^+$
Initialisation : $w_0 \longleftarrow 0$; $b_0 \longleftarrow 0$; $k \longleftarrow 0$;
$R \longleftarrow \max_{1 \leq i \leq n} \|x_i\|$;
repeat
  for $i = 1, \cdots, n$ **do**
    if $y_i(\langle w_k, x_i \rangle + b_k) \leq 0$ **then**
      $w_{k+1} \longleftarrow w_k + \eta y_i x_i$;
      $b_{k+1} \longleftarrow b_k + \eta y_i R^2$;
      $k \longleftarrow k + 1$;
    end
  end
until *aucune erreur pour une itération de la boucle*;
Result: l'hyperplan (w_k, b_k)

Hyperparamètre

Le taux d'apprentissage η.
Si sa valeur est trop petit l'algorithme mettra longtemps à converger. Si sa valeur est trop grande, l'algorithme ne convergera pas et oscillera entre plusieurs solutions.

Algorithme du Perceptron (dans le dual)

L'idée

On remarque que dans la stratégie du perceptron on ajoute à chaque itération un $y_i x_i$ pondéré par un certain terme. On a donc : $w = \sum_{i=1}^{n} \alpha_i y_i x_i$.

Data: Un ensemble $\mathcal{S}$ d'apprentissage linéaire séparable et un taux $\eta \in \mathbb{R}^+$
Initialisation : $\alpha \longleftarrow 0$; $b_0 \longleftarrow 0$;;
$R \longleftarrow \max_{1 \leq i \leq n} \|x_i\|$;
repeat
  for $i = 1, \cdots, n$ **do**
    if $y_i(\langle \sum_{j=1}^{i-1} \alpha_j y_j x_j, x_i \rangle + b) \leq 0$ **then**
      $\alpha_i \longleftarrow \alpha_i + \eta$;
      $b \longleftarrow b + \eta y_i R^2$;
    end
  end
until *aucune erreur pour une itération de la boucle*;
Result: le couple (α, b)

Le classifieur final

Le classifieur final

$$h(x) = \text{sgn}(\langle w, x \rangle + b) \quad (1)$$

$$= \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i \langle x_i, x \rangle + b\right) \quad (2)$$

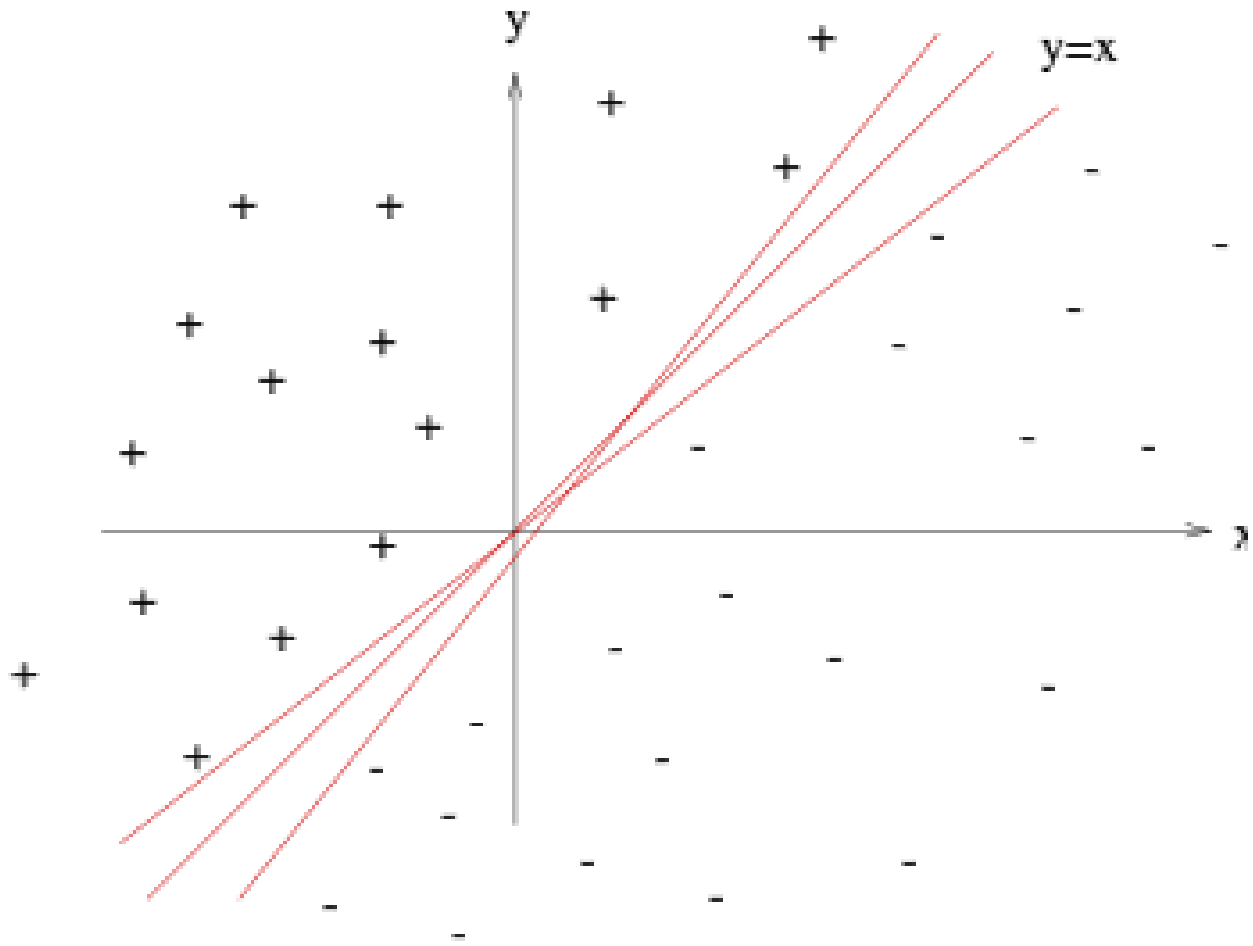
La matrice de Gram

$$G = [\langle x_i, x_j \rangle]_{i,j}$$

$\Longrightarrow$ On peut calculer la valeur du classifieur finale en utilisant les produits scalaires avec les exemples de la base d'apprentissage.

Sommaire

Le choix d'un hyperplan, un problème dont la solution n'est pas unique.



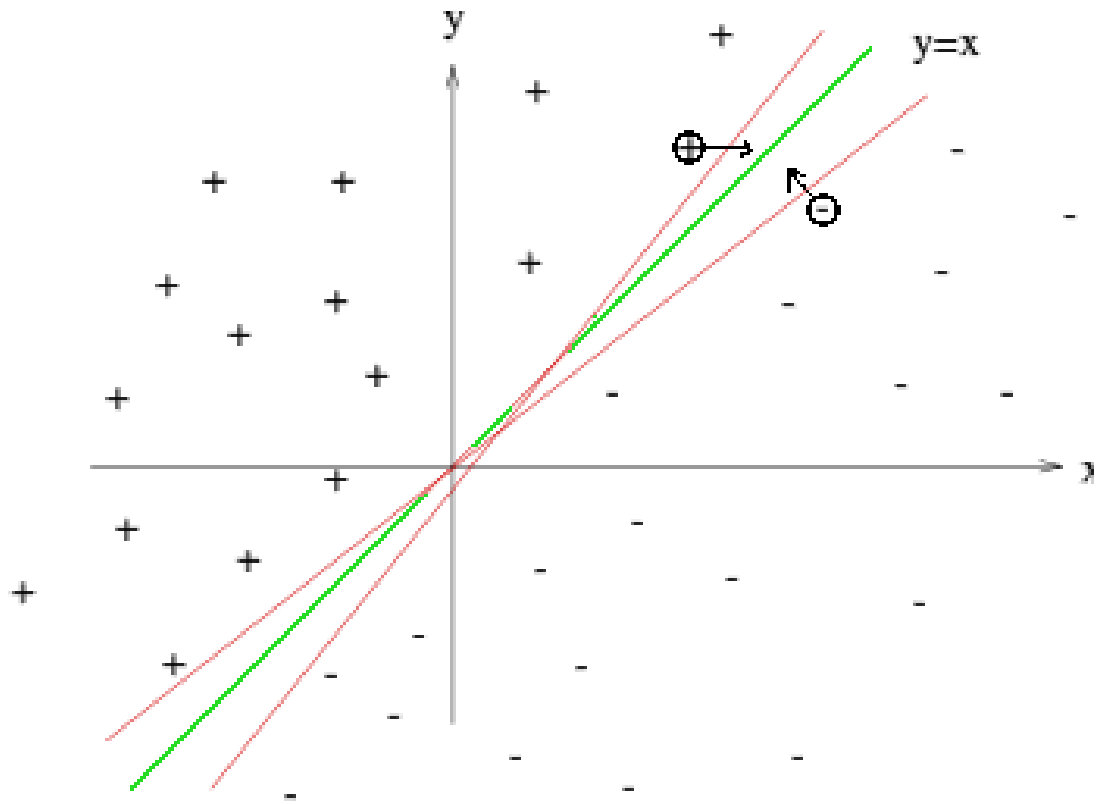
$\Longrightarrow$ Lequel choisir ? Quel est le meilleur ? Le perceptron est-il la meilleur solution ?

Quelles propriétés pour un bon hyperplan ?

Un bon hyperplan

Un bon hyperplan doit :

- correctement classer les données d'apprentissage,
- être robuste à des petites variations des données d'apprentissage.



La notion de marge

La marge

On définit la marge comme la distance entre l'hyperplan et son point d'apprentissage le plus proche.

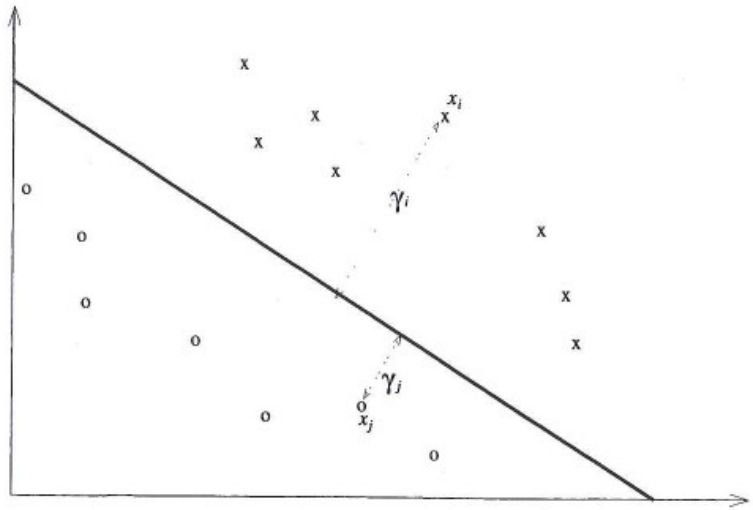


Figure 2.2: The geometric margin of two points

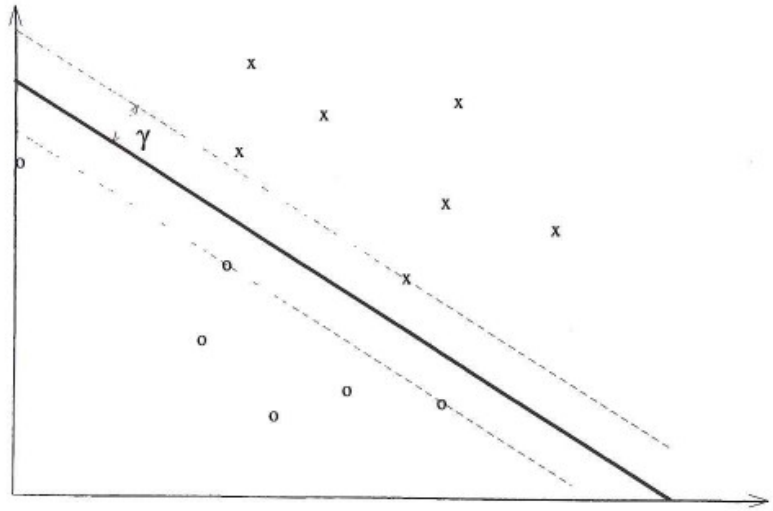
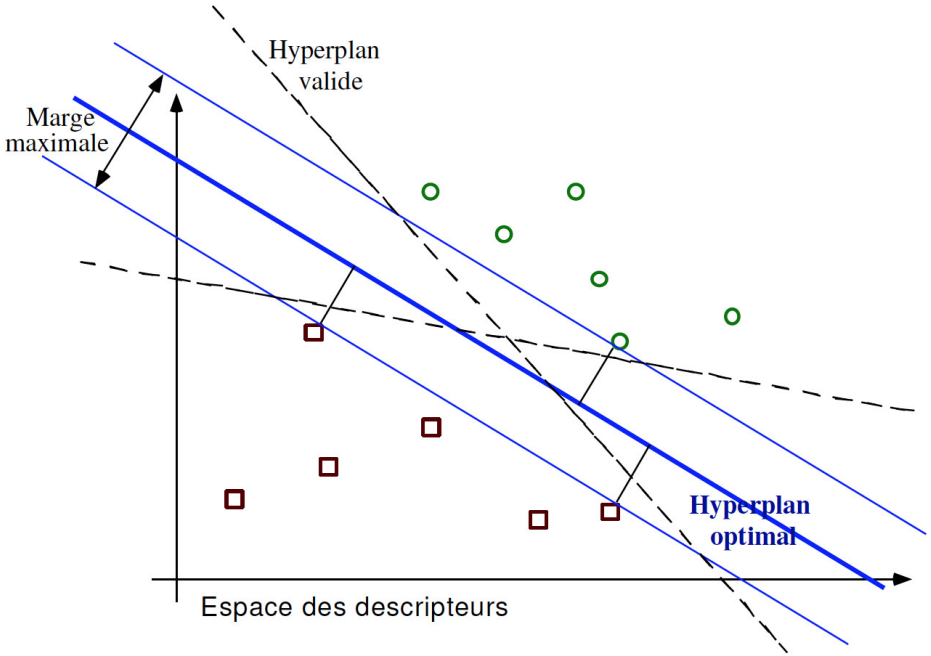


Figure 2.3: The margin of a training set

Où se trouve le meilleur hyperplan ?

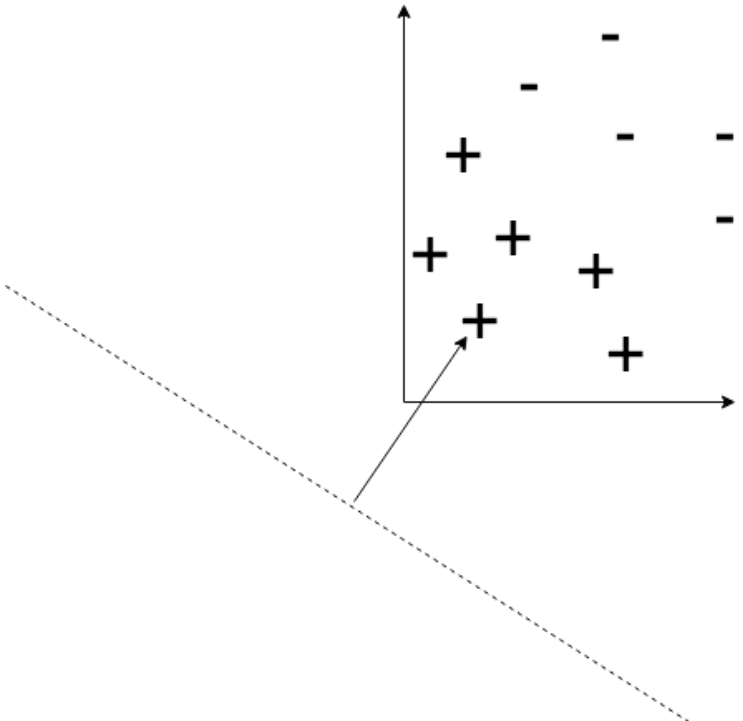
Stabilité de l'hyperplan aux variations des exemples : Maximiser la marge

Maximiser la marge de l'hyperplan permet de tolérer le maximum possible de variation des exemples d'apprentissages.



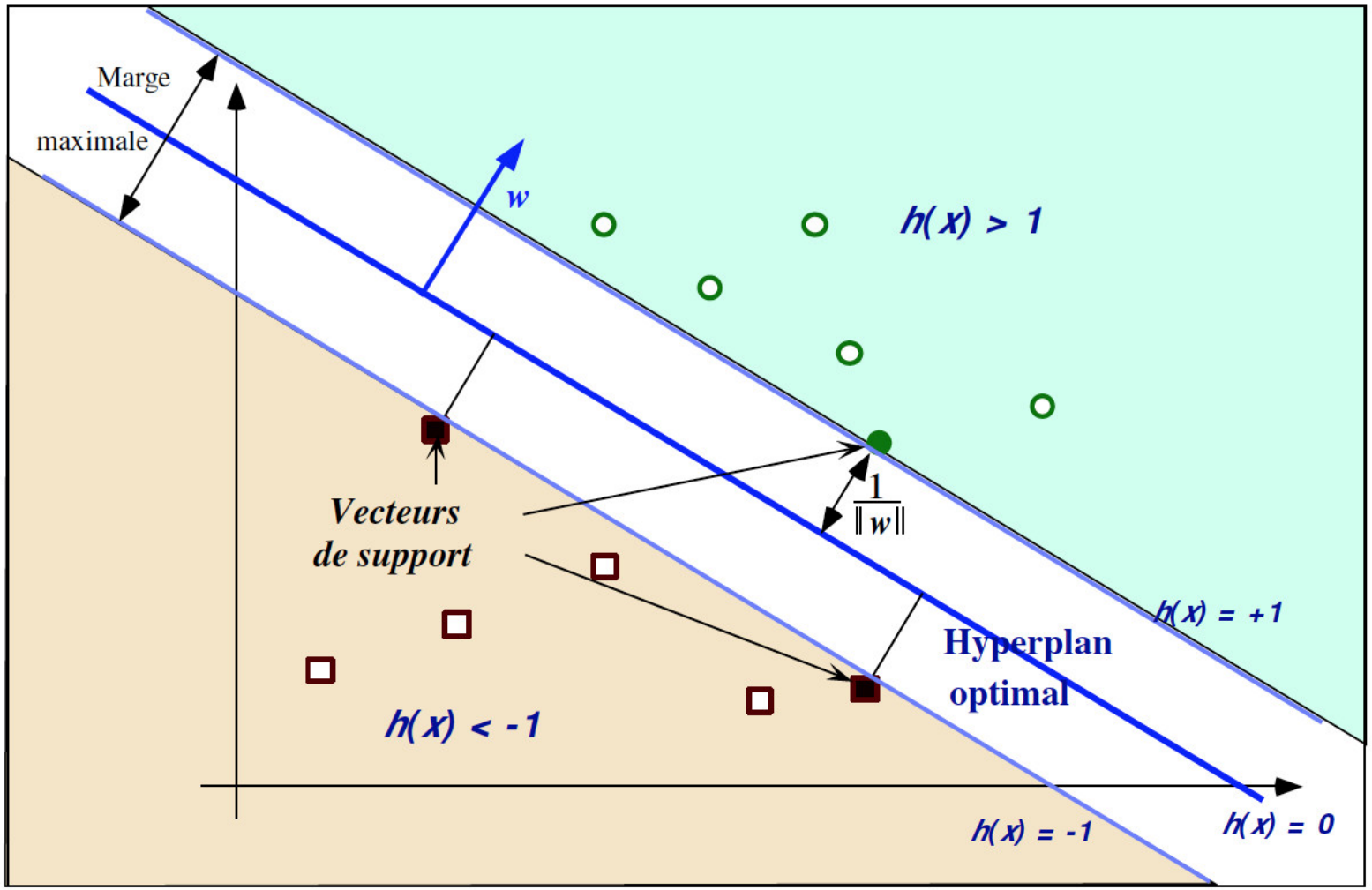
Maximiser la marge : oui mais ...

Attention à ne pas oublier de bien classer les exemples d'apprentissages $\Longrightarrow$ les marges entre les deux classes sont égales.



L'hyperplan maximisant la marge

La fonction de décision h est invariante par changement d'échelle, on peut donc choisir notre hyperplan sous une forme canonique :



Résolution du problème d'optimisation du SVM

Problème d'optimisation du SVM dans le primal

$$\arg\min_{w,b} \quad \frac{1}{2}\|w\|^2 \tag{3}$$

$$\text{sous contraintes} \quad \forall i, y_i(w^\top x_i + b) \geq 1 \tag{4}$$

$\Longrightarrow$ c'est un problème d'optimisation quadratique avec contrainte. La littérature mathématique nous donnes des solutions pour ce type de problème classique (algorithme du gradient conjugué, algorithme du simplexe, algorithme des points intérieurs...).

Résolution en pratique

- SVM-Light de Joachim,
- Sequential Minimal Optimisation (SMO) de Platt,
- libSVM,
- LaSVM,
- quadprog de Matlab...

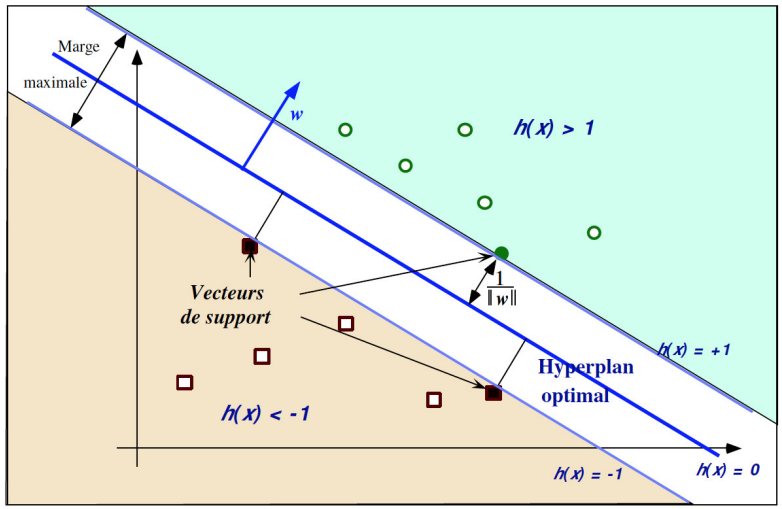
Les vecteurs supports

Les vecteurs supports

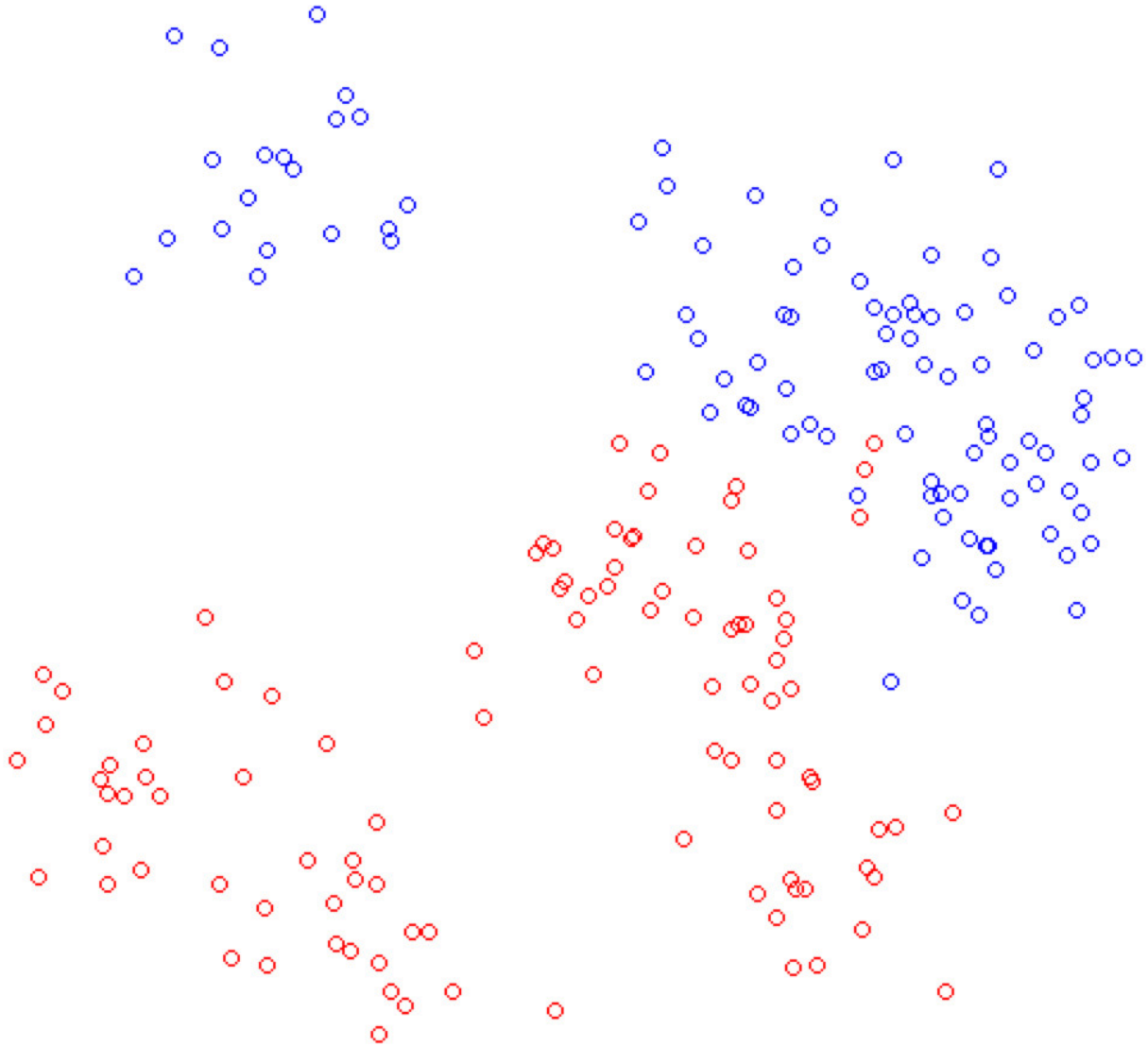
- Seuls les vecteurs supportant la marge définissent l'hyperplan.
- Par conséquence $w = \sum_i \alpha_i y_i x_i$ avec $\alpha_i = 0$ pour les exemples qui ne sont pas supports.

Problème d'optimisation du SVM dans le dual

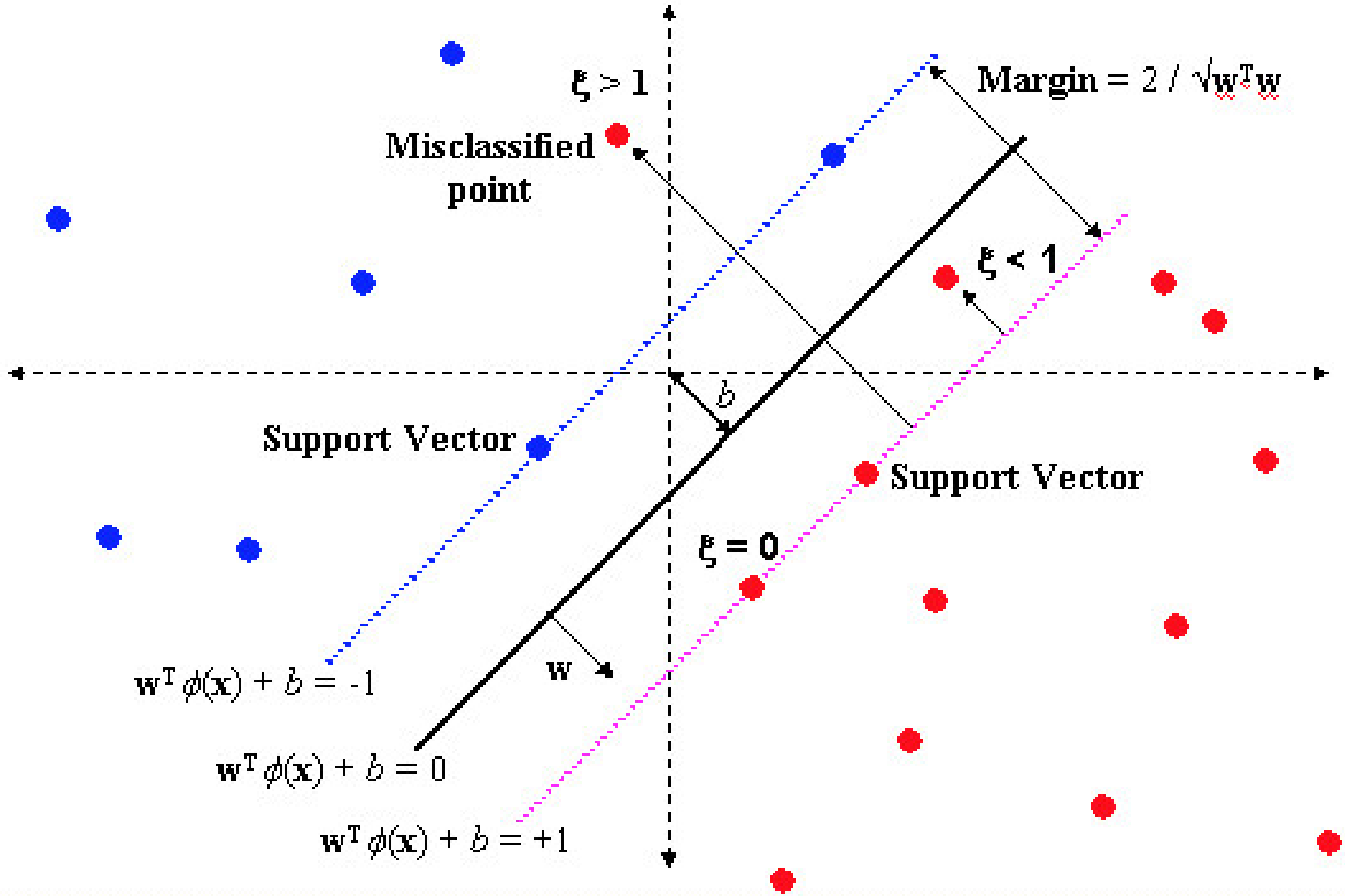
$$\begin{aligned} \arg\max_\alpha \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{s.t} \quad & \alpha_i \geq 0 \quad \forall i \\ & \sum_i \alpha_i y_i = 0 \end{aligned}$$



Comment faire dans un cas imparfaitement linéairement séparable ?



Relâcher la contrainte, utiliser des variables ressorts



SVM à marge souple

Principe

Dans le cas non linéairement séparable et à condition de rechercher une séparation linéaire, on introduit des variables d'écart :

$$y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i$$

Les anciens contraintes sont violés si

- (x_i, y_i) est dans la bande de séparation de l'hyperplan mais est du bon côté
- (x_i, y_i) est du mauvais côté de l'hyperplan

$\Longrightarrow$ le but est de minimiser les erreurs de classification $\sum_i 1_{\xi>0}$. Cette fonction étant non continue et non différentiable, on l'approxime par $\sum_i \xi_i$.

SVM à marge souple

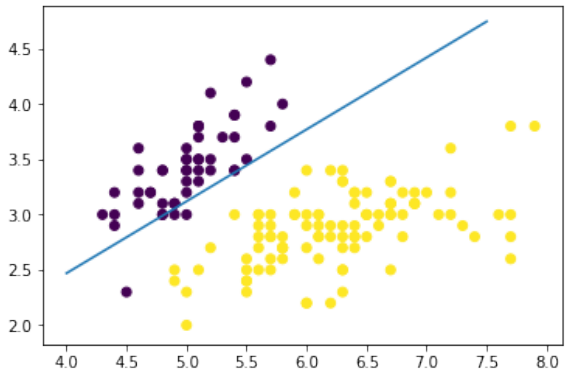
SVM à marge souple dans le primal

$$\arg\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \tag{5}$$

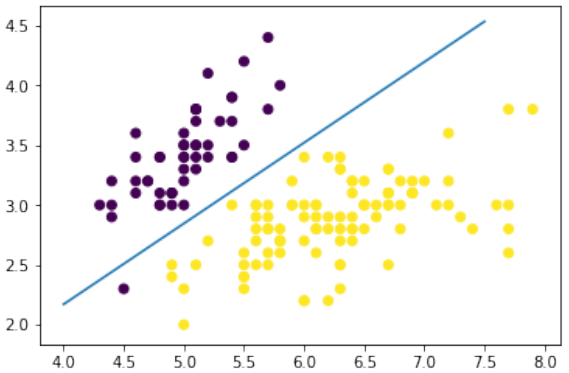
$$\text{s.t.} \quad \forall i, y_i(w^\top x_i + b) \geq 1 - \xi_i \tag{6}$$

Interprétation

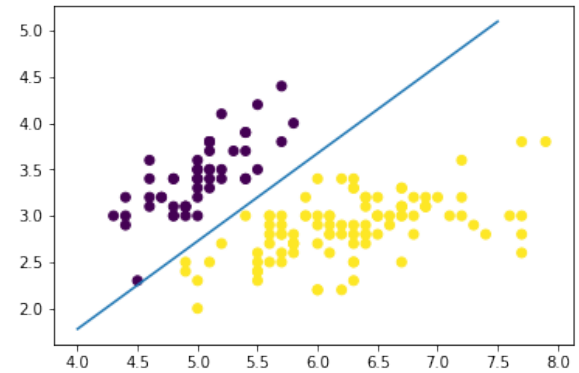
On autorise le relâchement de certaines contraintes pour résoudre le problème mais on en veux le moins possible. Par conséquence, on pénalise les relâchements.
Cela revient à ajouter un terme de régularisation de la solution. On peut ainsi régler le compromis entre maximisation de la marge et respect des contraintes.



C=0.01



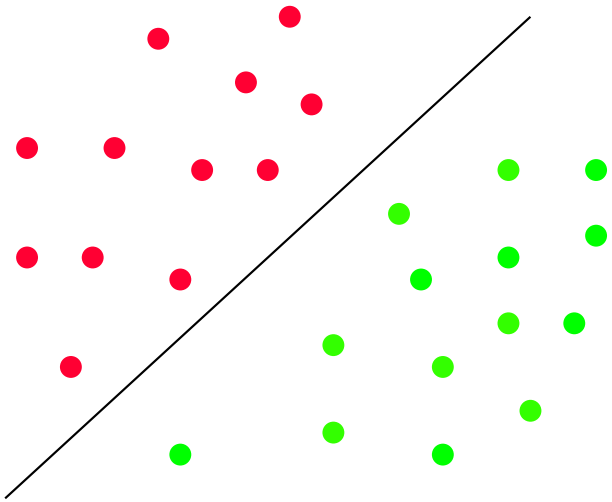
C=1



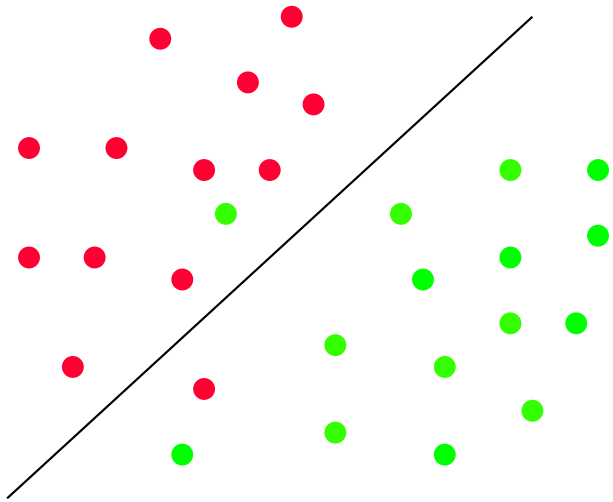
C=100

Sommaire

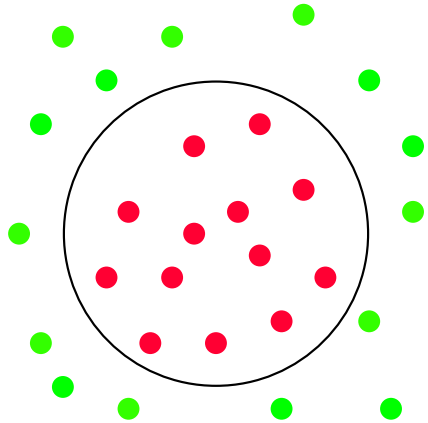
Différent catégories de problème de classification



Problème linéaire $\Longrightarrow$ OK

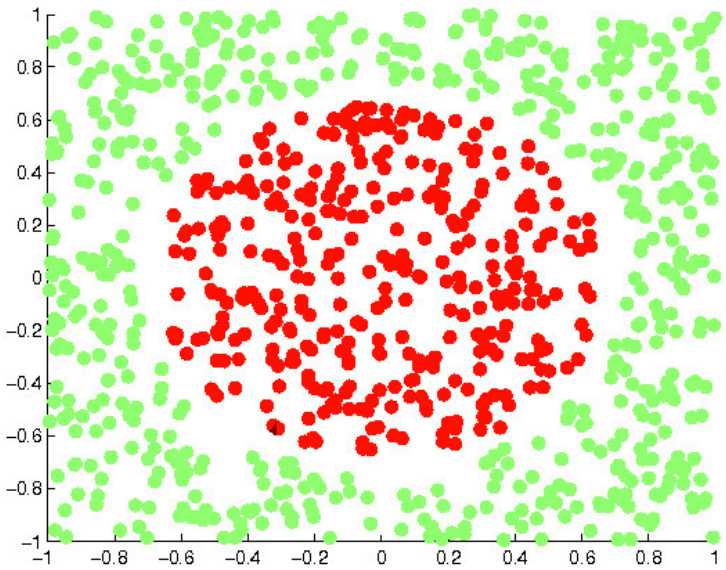


Problème quasi-linéaire $\Longrightarrow$ OK

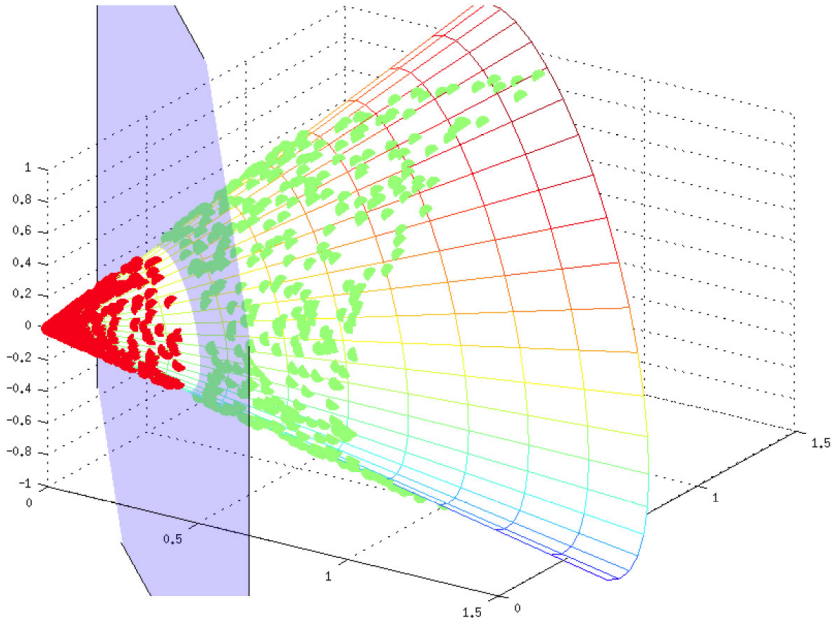


Problème non-linéaire ?

Astuce pour résoudre un problème non-linéaire : changer d'espace



Avant



Après

Formulation mathématique de notre exemple

Changement d'espace

$$\varphi : (x_1, x_2) \longmapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

Remarques

- En 2D, une surface de séparation non-linéaire est nécessaire pour séparer les données.
- En 3D, un hyperplan suffit.

Cas général

Théorème de Cover (1965)

- Dans un espace de dimension d, la probabilité que deux classes quelconques de n exemples ne soit pas linéairement séparable tend vers 0 lorsque $d \longrightarrow \infty$.
- Si $n < d$, il existe un hyperplan séparant les exemples d'apprentissages.

$\Longrightarrow$ Si on augmente suffisamment le nombre de dimension, on peut se ramener à un problème de classification linéaire.

$\Longrightarrow$ On va transformer l'espace d'entrée en un espace de plus grande dimensions.

Séparation non linéaire

Principe

1. On projette les exemples dans un nouvel espace de dimension d. (d étant souvent très grand, éventuellement infini). Cet espace est appelé l'espace des caractéristiques.

$$\varphi(x) = \begin{pmatrix} \varphi_1(x) \\ \varphi_2(x) \\ \vdots \\ \varphi_d(x) \end{pmatrix}$$

2. On effectue la séparation linéaire dans ce nouvel espace.

$\Longrightarrow$ Attention si d devient trop grand (voir infini), les calculs peuvent devenir rapidement impossible numériquement.

Les fonctions noyaux

Définition

Soit $\varphi : \mathcal{X} \longrightarrow \mathcal{H}$ une fonction de changement d'espace à valeur dans un espace de Hilbert ($\mathcal{H}$), on définit la fonction noyau associé à cette espace comme étant la fonction bilinéaire $k(x_1, x_2) : \mathcal{X} \times \mathcal{X} \longrightarrow \mathcal{H}$, vérifiant

$$k(x_1, x_2) = \langle \varphi(x_1), \varphi(x_2) \rangle$$

Remarque

Le produit scalaire est une fonction noyau. Elle correspond à une fonction noyau associé à la fonction $\varphi(x) = x$.

Retour sur notre exemple

Retour sur notre exemple initial

$$\varphi : (x_1, x_2) \longmapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

Calcul de la fonction noyau associé

$$\begin{aligned} \langle \varphi(x), \varphi(z) \rangle &= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \rangle & (7) \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 & (8) \\ &= (x_1z_1 + x_2z_2)^2 & (9) \\ &= \langle x, z \rangle^2 & (10) \end{aligned}$$

Remarque

- Il n'est pas nécessaire de passer dans l'espace de redescription pour calculer la valeur de la fonction noyau. Le calcul peut etre fait dans l'espace initial.

Autres exemples de fonctions noyaux

Quelques noyaux usuels

- Noyau linéaire : $k(x_i, x_j) = \langle x_i, x_j \rangle$.
- Noyaux polynomiaux : $k(x_i, x_j) = (\langle x_i, x_j \rangle + c)^q$, avec $c \in \mathbb{R}$ et $q \in \mathbb{R}^+$.
- Noyaux Gaussiens : $k(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$, $\sigma \in \mathbb{R}^+$.
- ...

Le noyau gaussien

Définition du noyau gaussien

$$k(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}, \sigma \in \mathbb{R}^+$$

Interprétation

Si $\|x_i - x_j\| \ll \sigma$ alors $k(x_i, x_j) \simeq 1 - \frac{\|x_i - x_j\|^2}{2\sigma^2}$, le noyau gaussien se comporte localement à peu près comme un "simple" produit scalaire.
Si $\|x_i - x_j\| \gg \sigma$ alors $k(x_i, x_j) \simeq 0$, la valeur du noyau est à peu près nulle quelque soit les données.

L'astuce du noyau

L'astuce du noyau

Un algorithme utilisant un produit scalaire peut être "noyauté" en remplaçant tous ces produits scalaires par une fonction noyau.

Étape d'utilisation

1. Exprimer l'algorithme uniquement avec des produits scalaires.
2. Remplacer tout les produits scalaires par une fonction noyau.

L'algorithme du Perceptron dans le dual

L'algorithme du Perceptron

Data: Un ensemble $\mathcal{S}$ d'apprentissage linéaire séparable et un taux $\eta \in \mathbb{R}^+$
Initialisation : $\alpha \longleftarrow 0$; $b_0 \longleftarrow 0$;;
$R \longleftarrow \max_{1 \leq i \leq n} \|x_i\|$;
repeat
  for $i = 1, \cdots, n$ **do**
    if $\sum_{j=1}^{i-1} \alpha_j y_i y_j \langle x_j, x_i \rangle + b \leq 0$ **then**
      $\alpha_i \longleftarrow \alpha_i + \eta$;
      $b \longleftarrow b + \eta y_i R^2$;
    end
  end
until *aucune erreur pour une itération de la boucle*;
Result: le couple (α, b)

Version "noyauté" de l'algorithme du Perceptron

L'algorithme du Perceptron

Data: Un ensemble $\mathcal{S}$ d'apprentissage linéaire séparable et un taux $\eta \in \mathbb{R}^+$
Initialisation : $\alpha \longleftarrow 0$; $b_0 \longleftarrow 0$;;
$R \longleftarrow \max_{1 \leq i \leq n} \|x_i\|$;
repeat
  for $i = 1, \cdots, n$ **do**
    if $\sum_{j=1}^{i-1} \alpha_j y_i y_j k(x_j, x_i) + b \leq 0$ **then**
      $\alpha_i \longleftarrow \alpha_i + \eta$;
      $b \longleftarrow b + \eta y_i R^2$;
    end
  end
until *aucune erreur pour une itération de la boucle*;
Result: le couple (α, b)

Rappel : Le classifieur final dans le dual

Le classifieur final

$$h(x) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i \langle x_i, x \rangle + b\right) \tag{11}$$

La matrice de Gram

$$G = [\langle x_i, x_j \rangle]_{i,j}$$

$\Longrightarrow$ On peut calculer la valeur du classifieur finale en utilisant les produits scalaires avec les exemples de la base d'apprentissage.

Le classifieur final avec l'astuce du noyau

Le classifieur final

$$h(x) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b\right) \quad (12)$$

La matrice de Gram

$$G = [k(x_i, x_j)]_{i,j}$$

$\Longrightarrow$ On peut calculer la valeur du classifieur finale en utilisant les produits scalaires avec les exemples de la base d'apprentissage.

Application au SVM

SVM linéaire dans le dual

$$\arg\max_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{13}$$

$$\text{s.t} \quad \forall i \quad C \geq \alpha_i \geq 0 \tag{14}$$

$$\sum_i \alpha_i y_i = 0 \tag{15}$$

SVM non-linéaire dans le dual

$$\arg\max_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{16}$$

$$\text{s.t} \quad \forall i \quad C \geq \alpha_i \geq 0 \tag{17}$$

$$\sum_i \alpha_i y_i = 0 \tag{18}$$

Conseils de lecture pour aller plus loin

- The Nature of Statistical Learning Theory. Vladimir Vapnik
- An introduction to support Vector Machines and other kernel-based learning methods. Nello Cristianini and John Shawe-Taylor
- Learning with Kernels. Bernhard Schölkopf and Alexander J.Smola.
- Deep Learning, Ian Goodfellow, Yoshua Bengio and Aaron Courville.
- Convex Optimization. Stephen Boyd and Lieven Vandenberghe.