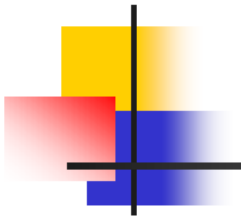
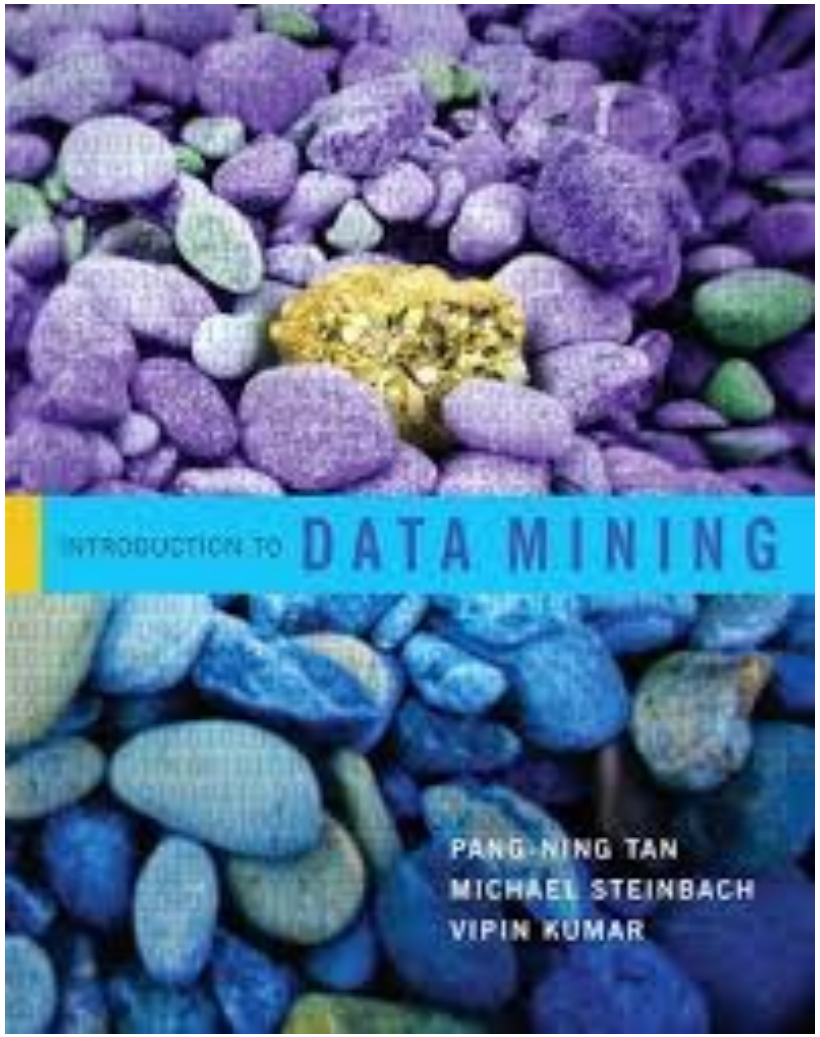


INF7370

Support Vector Machine

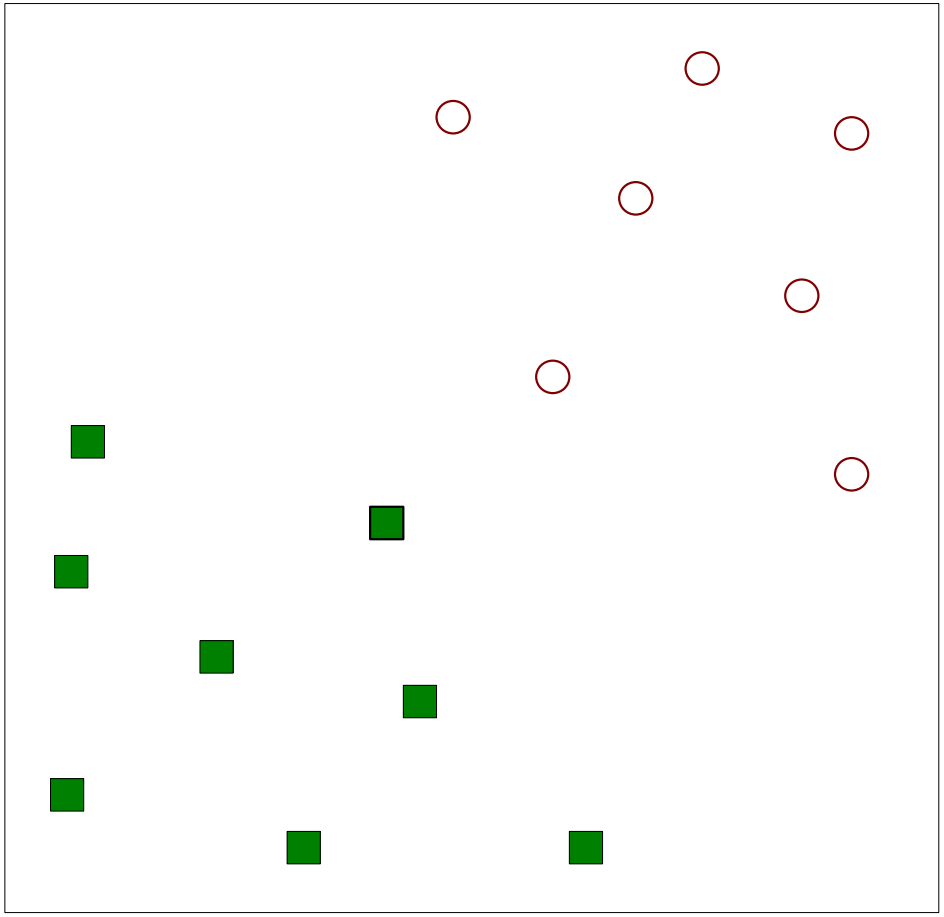
bouguessa.mohamed@uqam.ca

Sources


INTRODUCTION TO DATA MINING
PANG-NING TAN
MICHAEL STEINBACH
VIPIN KUMAR

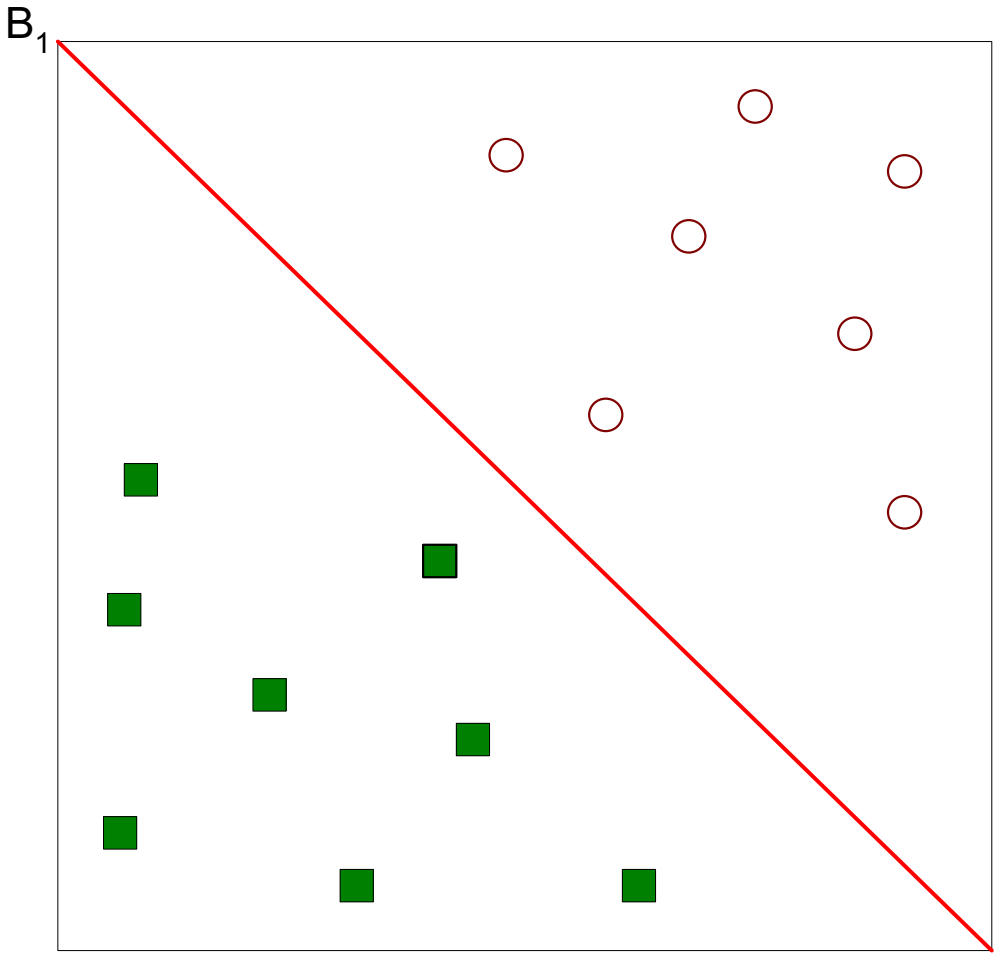

Algorithmes
3e édition
Antoine Cornuéjols
Laurent Miclet - Vincent Barra
Apprentissage artificiel
Deep learning, concepts et algorithmes
EYROLLES

Exemple



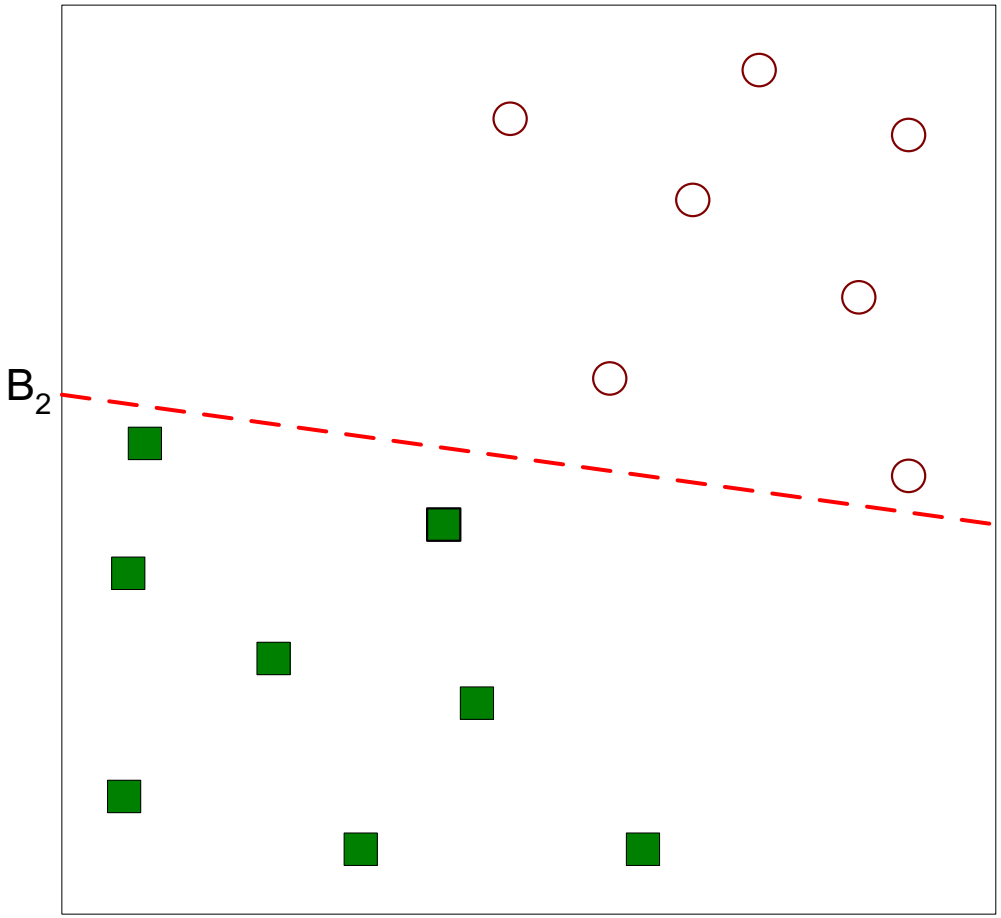
Identifier un hyperplan qui sépare les deux classes.

Exemple



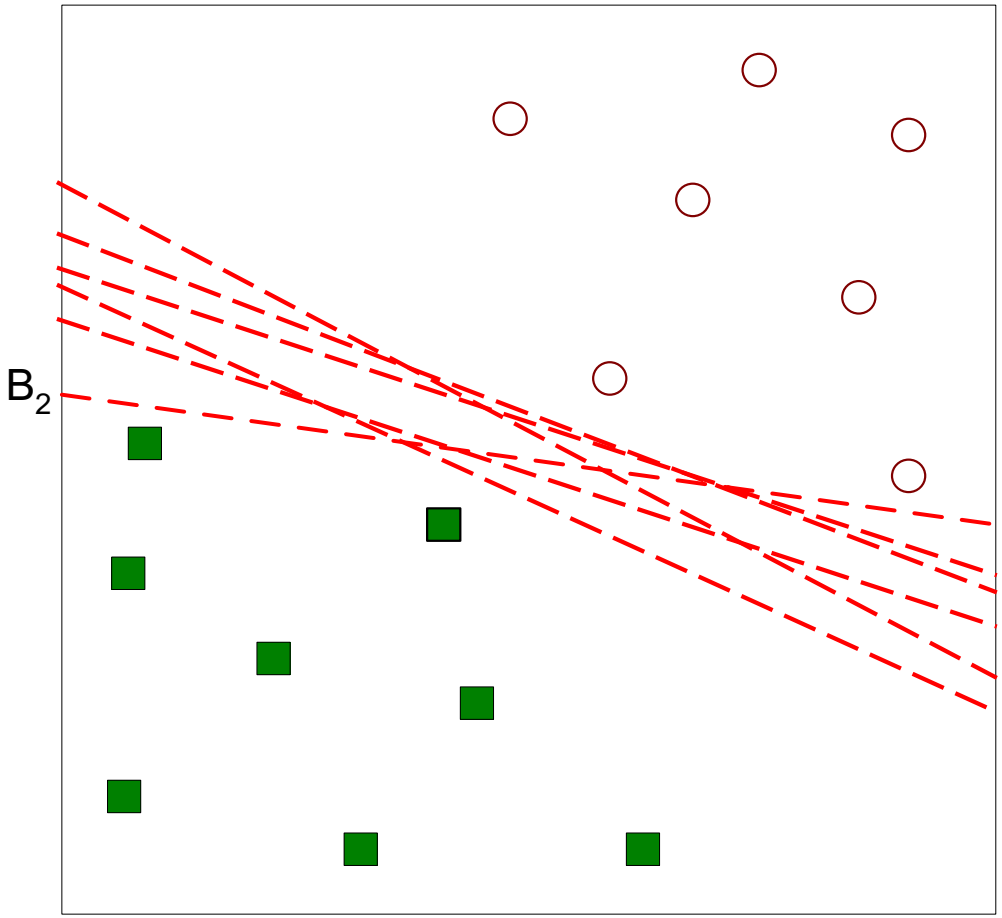
Une solution possible.

Exemple



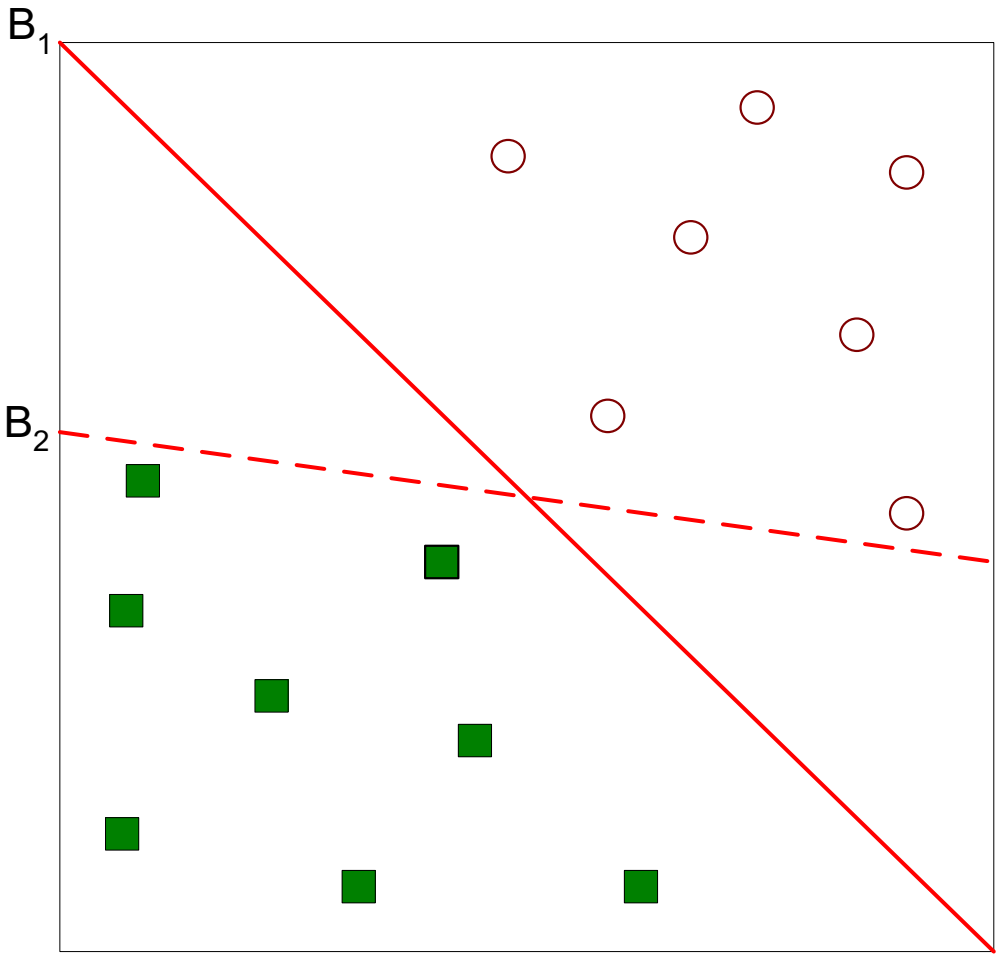
Une autre solution possible.

Exemple



Plusieurs solutions possibles.

Exemple



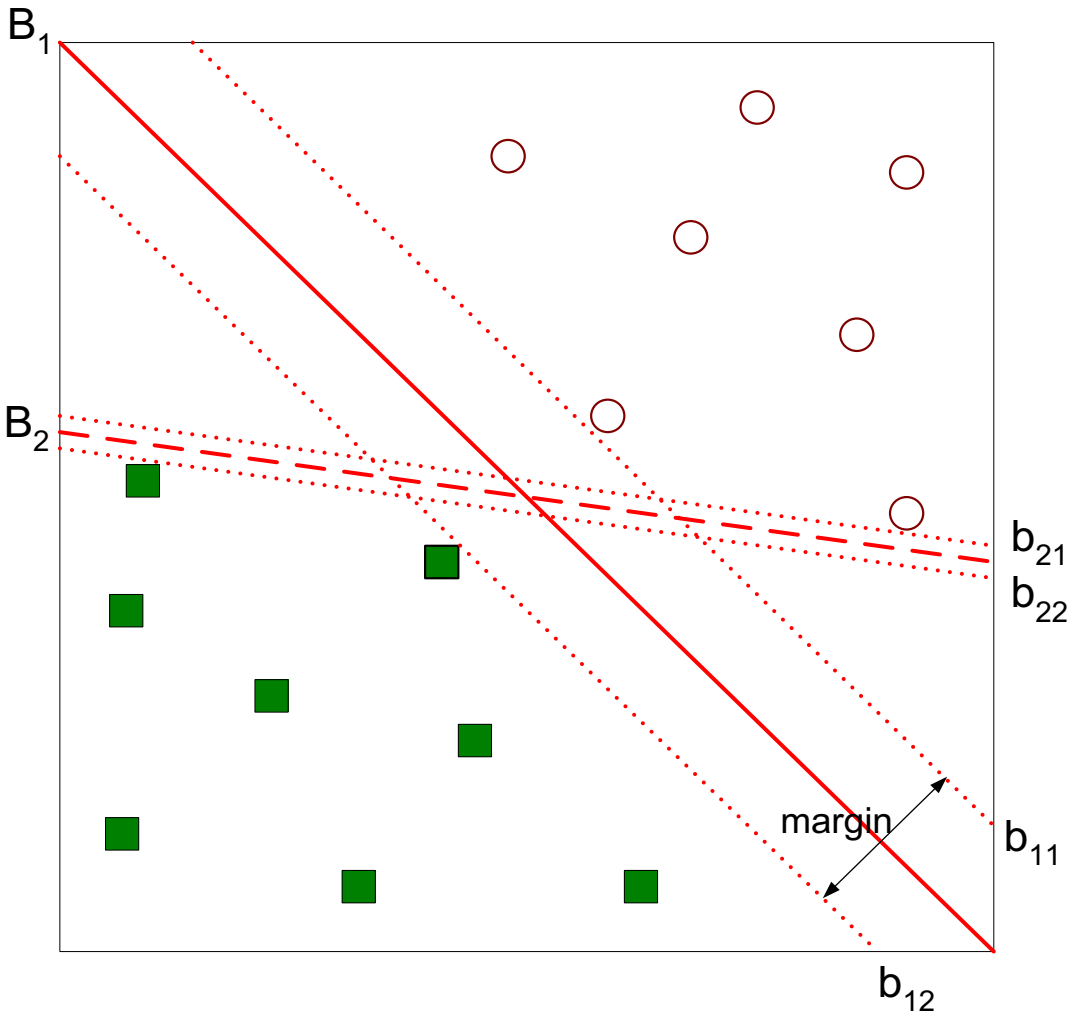
Quelle est la meilleure solution B1 ou B2?

Le meilleur séparateur linéaire

- Lorsqu'il existe une surface séparatrice linéaire entre les points d'apprentissage, il en existe plusieurs autres. On peut alors chercher parmi ces séparatrices celle qui est la meilleure : la plus écartée des nuages de points.

→ Séparateur à vaste marge

Séparateur à vaste marge

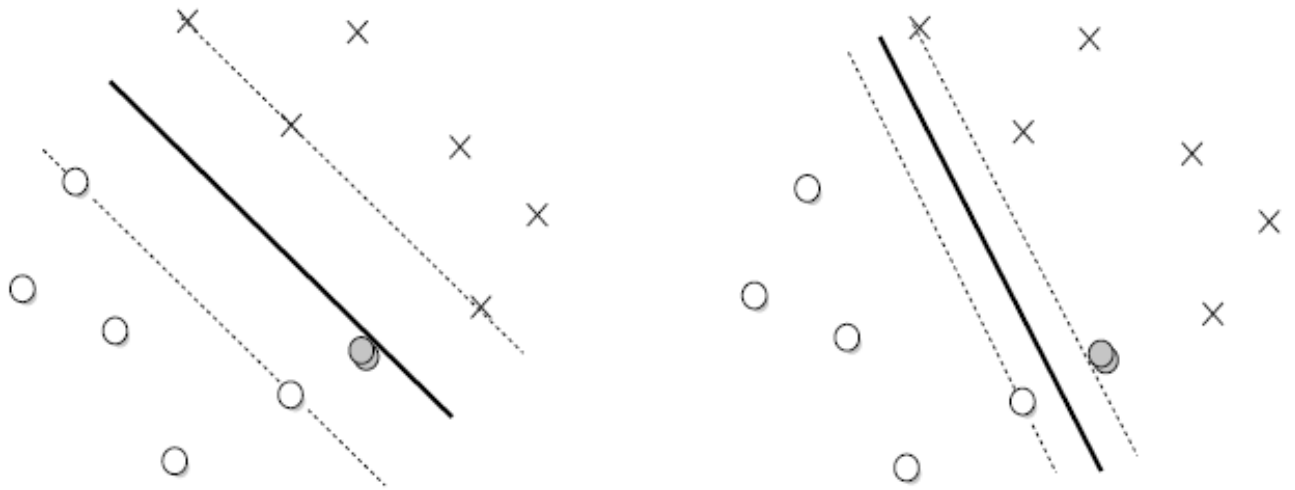


Trouver l'hyperplan qui maximise la marge.

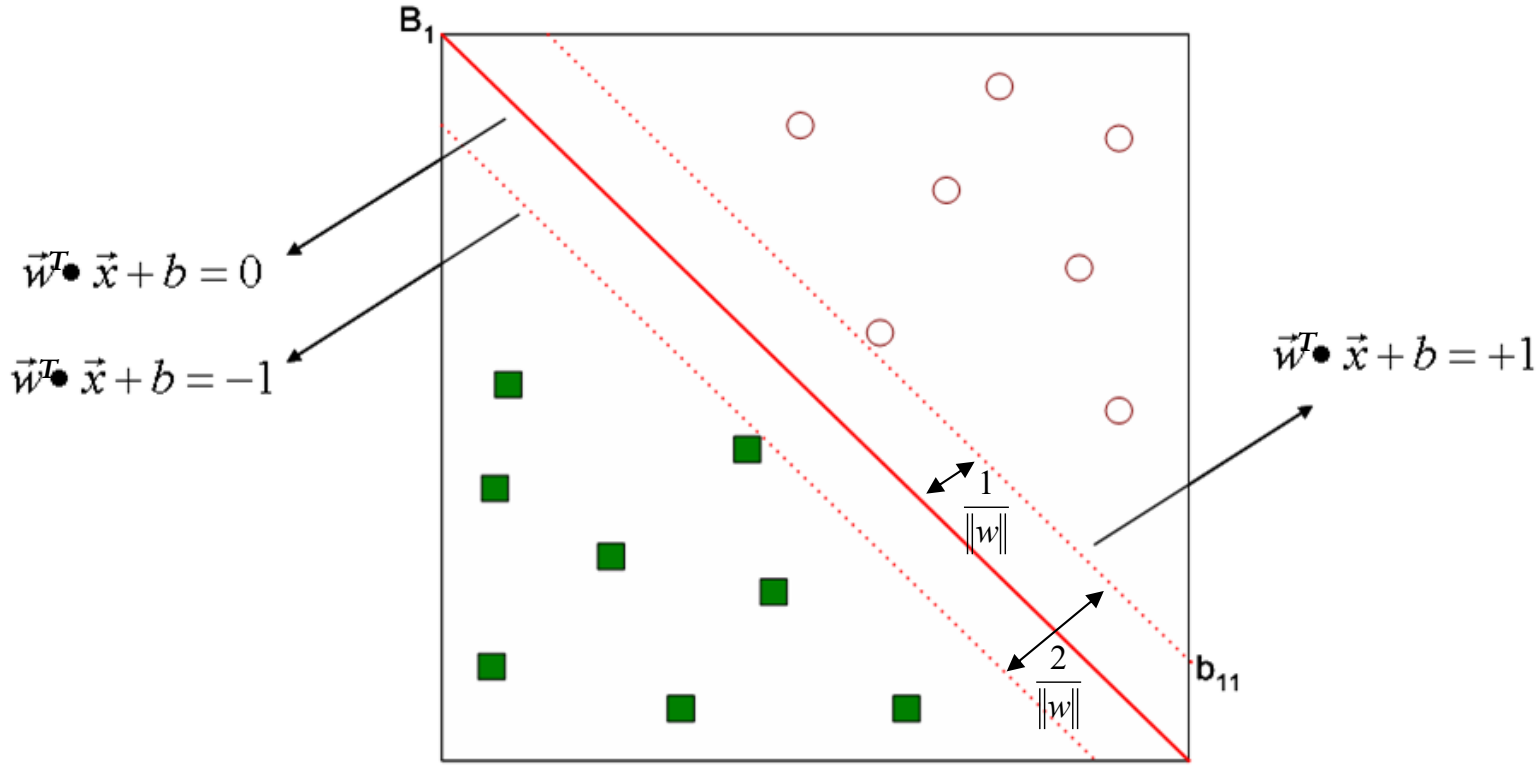
➔ B1et meilleur que B2

Séparateur à vaste marge

Intuitivement, le fait d'avoir une marge plus large procure plus de « sécurité » lorsqu'on classe un exemple inconnu. La partie gauche de la figure ci-dessous montre qu'avec l'hyperplan « optimal », un nouvel exemple reste bien classé (bien qu'il tombe dans la marge). On constate sur la partie droite qu'avec une plus petite marge, l'exemple est mal classé.



SVM linéaire : le cas des classes séparables



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w}^T \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w}^T \bullet \vec{x} + b \leq -1 \end{cases}$$

L'apprentissage du SVM linéaire

- L'apprentissage consiste à l'estimation des paramètres w et b.
- Ces deux paramètres doivent être choisis selon les deux conditions suivantes :

$$w^T \cdot x_i + b \geq 1 \, si \; y_i = 1,$$

$$w^T \cdot x_i + b \leq -1 \, si \; y_i = -1.$$

- Ces deux conditions imposent que tous les points d'apprentissage de la classe $y = 1$ doivent être situés sur ou au-dessus l'hyperplan $w^T . x + b = 1$, alors que les objets de la classe $y = -1$ doivent être situés sur ou au-dessous de l'hyperplan $w^T . x + b = -1$.
- Ces deux conditions peuvent être simplifiées comme

$$y_i(w^T \cdot x_i + b) \geq 1, \; i = 1, \cdots, n$$

L'expression primale du problème des SVM

- SVM impose aussi une autre condition : maximiser la marge.
- La maximisation de la marge est revient à minimiser la fonction objective suivante :

$$f(w) = \frac{\|w\|^2}{2}$$

→ L'apprentissage d'un SVM linéaire lorsque les classes sont séparables devient donc un problème d'optimisation de la fonction :

$$\min_w = \frac{\|w\|^2}{2}$$

avec la contrainte $y_i(w^T \cdot x_i + b) \geq 1, i = 1,2,\cdots,n.$

→ Cette écriture est appelée formulation primale.

L'expression duale du problème des SVM

- La fonction objective que nous voulons maximiser est quadratique et les contraintes sont exprimées par une équation linéaire en fonction de w et b → ce type de problème est connu dans la littérature comme un problème d'optimisation convexe.
- Une solution possible à ce problème d'optimisation est d'utiliser la méthode de multiplicateur de Lagrange qui incorpore la fonction objective et les contraintes dans une seule équation.
- On doit donc réécrire la fonction objective de telle sorte qu'elle tient compte des contraintes, mais dans une seule formule. La nouvelle fonction est connue sous le nom du Lagrangien.

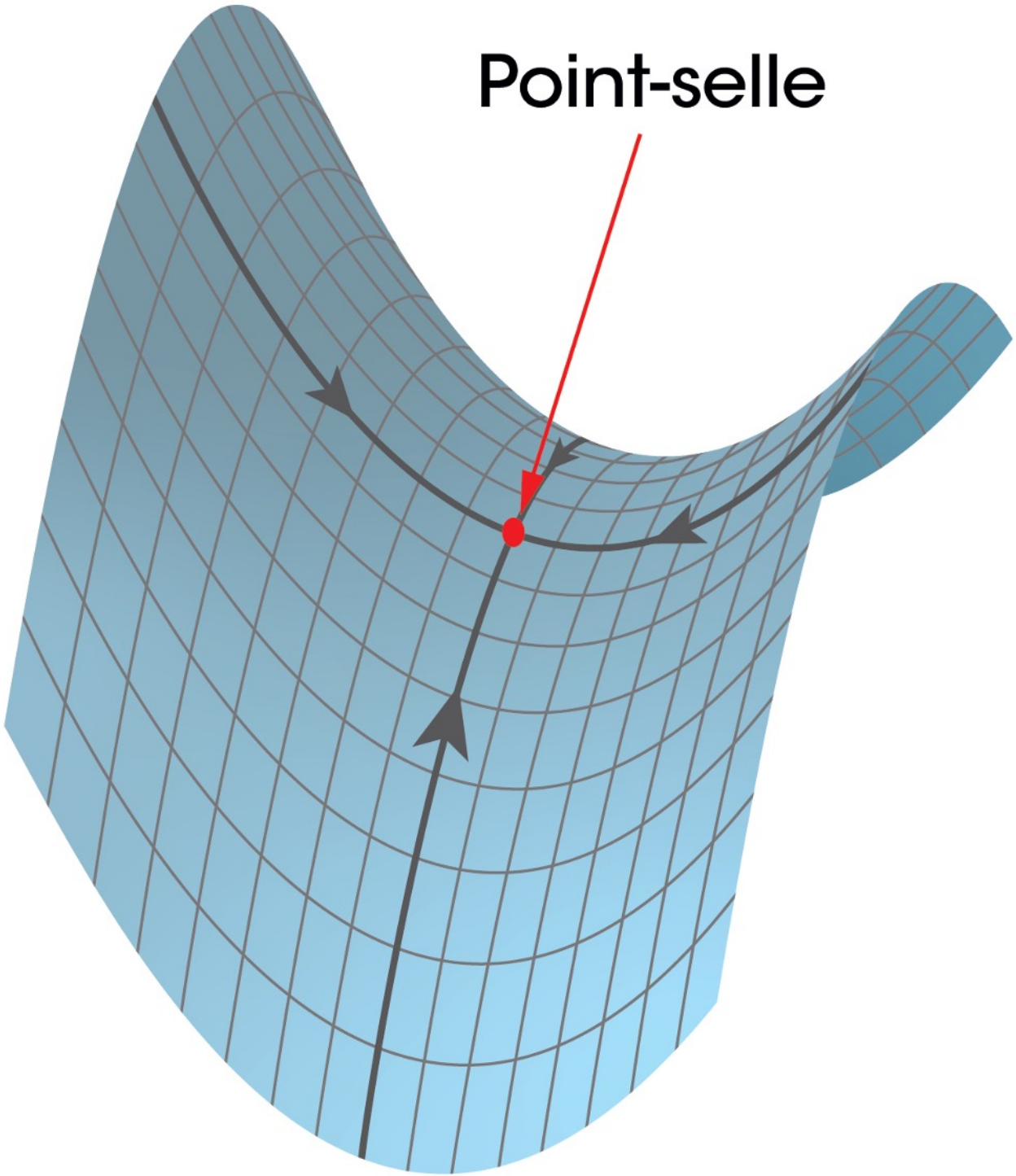
L'expression duale

- Le lagrangien est la somme de la fonction objective et d'une combinaison linéaire des contraintes.
- La nouvelle fonction à minimiser est donc:

$$L_P = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \lambda_i \left(y_i (w^T \cdot x_i + b) - 1 \right) \qquad (1)$$

- Les coefficients $\lambda_i \geq 0$ sont appelés multiplicateurs de Lagrange ou encore variables duales.

→ Le problème primal et sa formulation duale ont la même solution qui correspond à un point-selle de lagrangien (il faut le minimiser par rapport aux variables primaires w et b et le maximiser par rapport aux variables duales λ_i).


Point-selle

L'expression duale

- Afin de minimiser le Lagrangien, on doit estimer les dérivées de L_p par rapport à w et b et puis les mètres à zéro

$$L_P = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \lambda_i \left(y_i (w^T \cdot x_i + b) - 1 \right) \qquad (1)$$

$$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{n} \lambda_i y_i x_i, \qquad (2)$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n} \lambda_i y_i = 0. \qquad (3)$$

L'expression duale

- En substituant (2) et (3) dans (1), on élimine les variables primaires et l'on obtient la forme duale du problème d'optimisation :

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j (x_i^T \cdot x_j) \qquad (4)$$

- Avec la forme duale, on a un seul type de paramètres à estimer → λ_i
- Maintenant, le problème d'optimisation consiste à maximiser l'expression duale L_D illustrée par l'équation (4).

Solution du problème d'optimisation

Trouver les multiplicateurs de Lagrange λ_i tels que

$$\underset{\lambda}{Max}\left\{\sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{n}\lambda_i\lambda_j y_i y_j (x_i^T \cdot x_j)\right\} \qquad (5)$$

$$\lambda_i \geq 0, i = 1, \cdots, n$$

Ce problème est un problème d'optimisation quadratique. Les problèmes de ce type sont aussi regroupés sous l'appellation de programmation quadratique. On trouve beaucoup de solutions informatiques toutes prêtes pour résoudre ces problèmes de programmation quadratique, mettant en œuvre des techniques sortant du cadre de ce cours.

Voici un exemple d'une ressource qui se rattachent à la programmation quadratique

https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

Solution du problème d'optimisation

- Une fois que les λ_i sont trouvés, les paramètres w et b de l'hyperplan solution peuvent êtres estimés à partir des équations (2) et (3).

L'hyperplan solution correspondant peut être écrit:

$$h(x) = w^T \cdot x + b = \left(\sum_{i=1}^{n} \lambda_i y_i x_i^T \cdot x \right) + b$$

car selon l'équation (2) $\quad w = \sum_{i=1}^{n} \lambda_i y_i x_i$

→ **Comment estimer le paramètre *b* ?**

Solution du problème d'optimisation

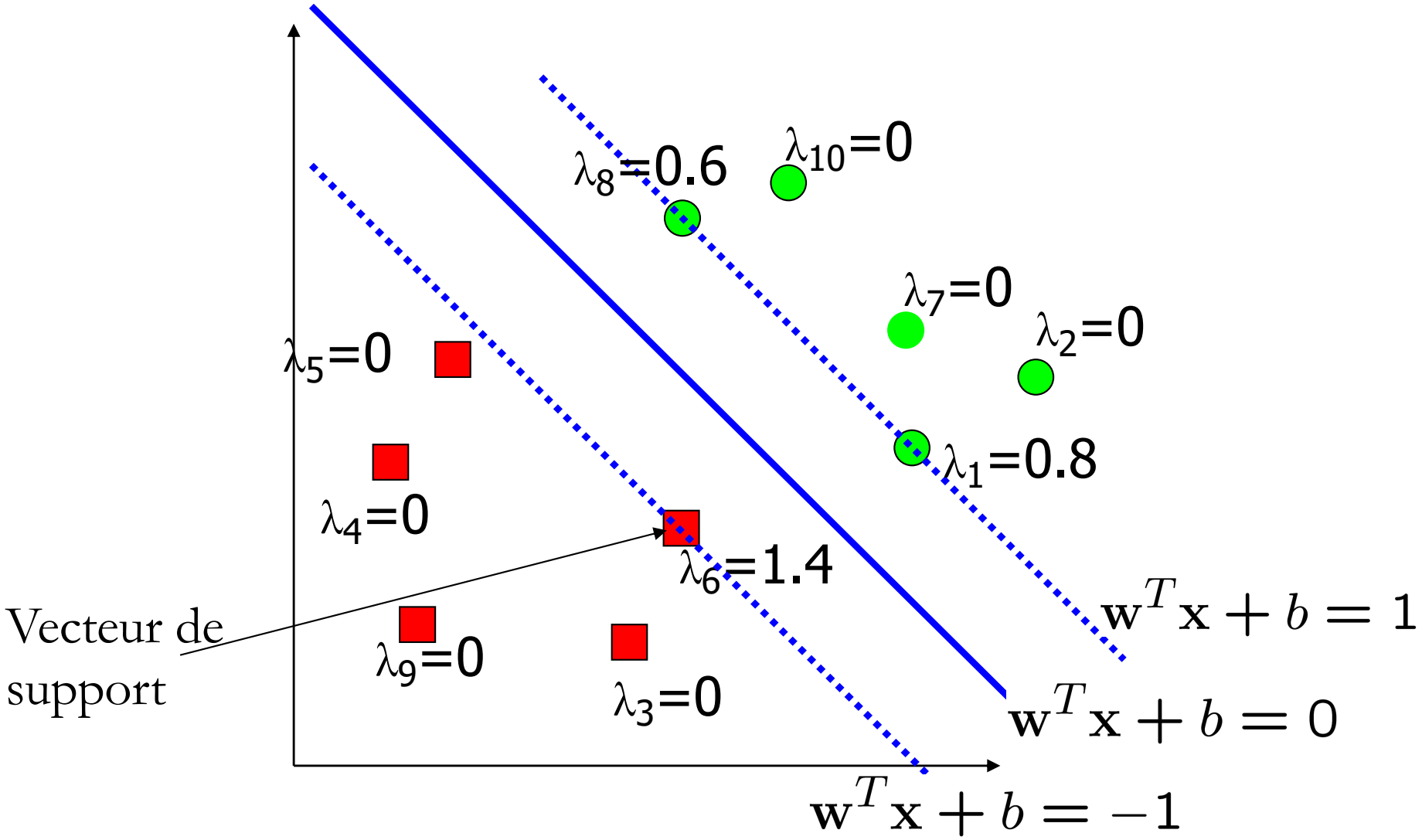
- Il est montré (conditions de Karush-Kuhn-Tucker (KKT)) que seuls les points qui sont sur les hyperplans frontière $w^T . x + b = \pm 1$ joue un rôle.
- Ces points sont appelés « vecteurs de support » par Vapnik
- Les multiplicateurs de Lagrange λ_i sont **non nuls seulement** pour ces points.

→ Le vecteur solution w a donc une expression en termes d'un sous-ensemble des exemples d'apprentissage :

→ **Les vecteurs de support.**

- C'est en même temps intuitivement satisfaisant puisque l'on voit bien que l'hyperplan solution est entièrement déterminé par ces exemples.

Illustration



Solution du problème d'optimisation

- La valeur de *b* est obtenue en utilisant n'importe quels exemples des données d'apprentissage qui représente un vecteur à support (points avec un multiplicateur de Lagrange non nul) dans l'équation suivante :

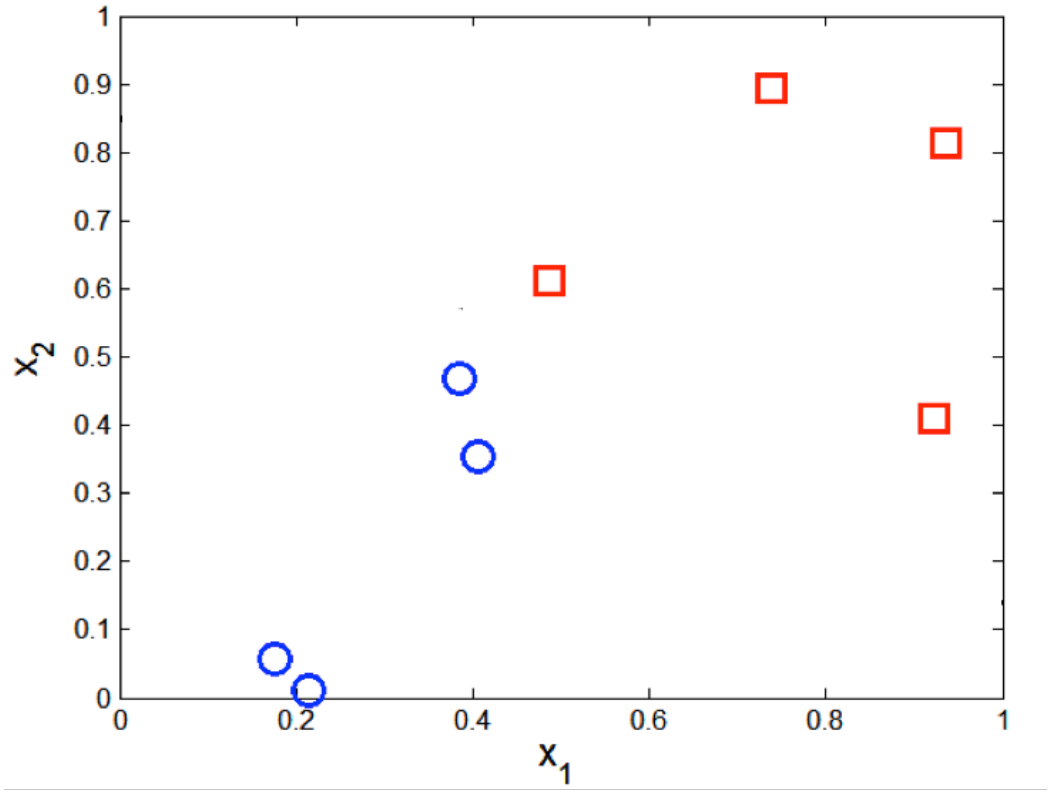
$$\lambda_i \left[y_i \left(w^T \cdot x_i + b \right) - 1 \right] = 0 \qquad (6)$$

- Remarque : comme il peut y avoir plusieurs vecteurs à support, il est recommandé de calculer la valeur de *b* pour chacun de ces points et puis estimer la moyenne comme la valeur finale de *b*.

Exemple

La figure suivante illustre un ensemble d'apprentissage qui contient 8 exemples répartis en deux classes. La figure montre aussi les multiplicateurs de Lagrange estimés pour chaque donnée d'apprentissage.

x1	x2	y	λ
0.3858	0.4687	-1	65.5261
0.4871	0.611	1	65.5261
0.9218	0.4103	1	0
0.7382	0.8936	1	0
0.1763	0.0579	-1	0
0.4057	0.3529	-1	0
0.9355	0.8132	1	0
0.2146	0.0099	-1	0



Exemple (suite)

x1	x2	y	λ
0.3858	0.4687	-1	65.5261
0.4871	0.611	1	65.5261
0.9218	0.4103	1	0
0.7382	0.8936	1	0
0.1763	0.0579	-1	0
0.4057	0.3529	-1	0
0.9355	0.8132	1	0
0.2146	0.0099	-1	0

w et b sont les paramètres
de l'hyperplan solution à estimer.

On a : $w = \sum_{i=1}^{n} \lambda_i y_i x_i$ donc

$$w_1 = \sum_{i=1}^{n} \lambda_i y_i x_{i1}$$

$$= (65.5261 \times (-1) \times 0.3858) + (65.5261 \times 1 \times 0.4871)$$

$$w_1 = 5.72609$$

$$w_2 = \sum_{i=1}^{n} \lambda_i y_i x_{i2}$$

$$= (65.5261 \times (-1) \times 0.4687) + (65.5261 \times 1 \times 0.611)$$

$$w_2 = 8.04366$$

Exemple (suite)

x1	x2	y	λ
0.3858	0.4687	-1	65.5261
0.4871	0.611	1	65.5261
0.9218	0.4103	1	0
0.7382	0.8936	1	0
0.1763	0.0579	-1	0
0.4057	0.3529	-1	0
0.9355	0.8132	1	0
0.2146	0.0099	-1	0

On a $w^T x + b = 1$ et $w^T x + b = -1$

- $w^T x + b^{(1)} = 1$ (classe +)

$\Rightarrow b^{(1)} = 1 - w^T x$

$= 1 - \left[(5.72609,\ 8.04366) \begin{pmatrix} 0.4871 \\ 0.611 \end{pmatrix} \right]$

$= 1 - \left[(5.72609 * 0.4871) + (8.04366 * 0.611) \right]$

$b^{(1)} = -7.10604$

Exemple (suite)

x1	x2	y	λ
0.3858	0.4687	-1	65.5261
0.4871	0.611	1	65.5261
0.9218	0.4103	1	0
0.7382	0.8936	1	0
0.1763	0.0579	-1	0
0.4057	0.3529	-1	0
0.9355	0.8132	1	0
0.2146	0.0099	-1	0

$\cdot\ w^T \cdot x + b = -1$ (classe −)

$b^{(2)} = -1 - w^T x$

$= -1 - \left[(5.72609, 8.04366) \begin{pmatrix} 0.3858 \\ 0.4687 \end{pmatrix} \right]$

$= -1 - \left[(5.72609 * 0.3858) + (8.04366 * 0.4687) \right]$

$b^{(2)} = -6.97919$

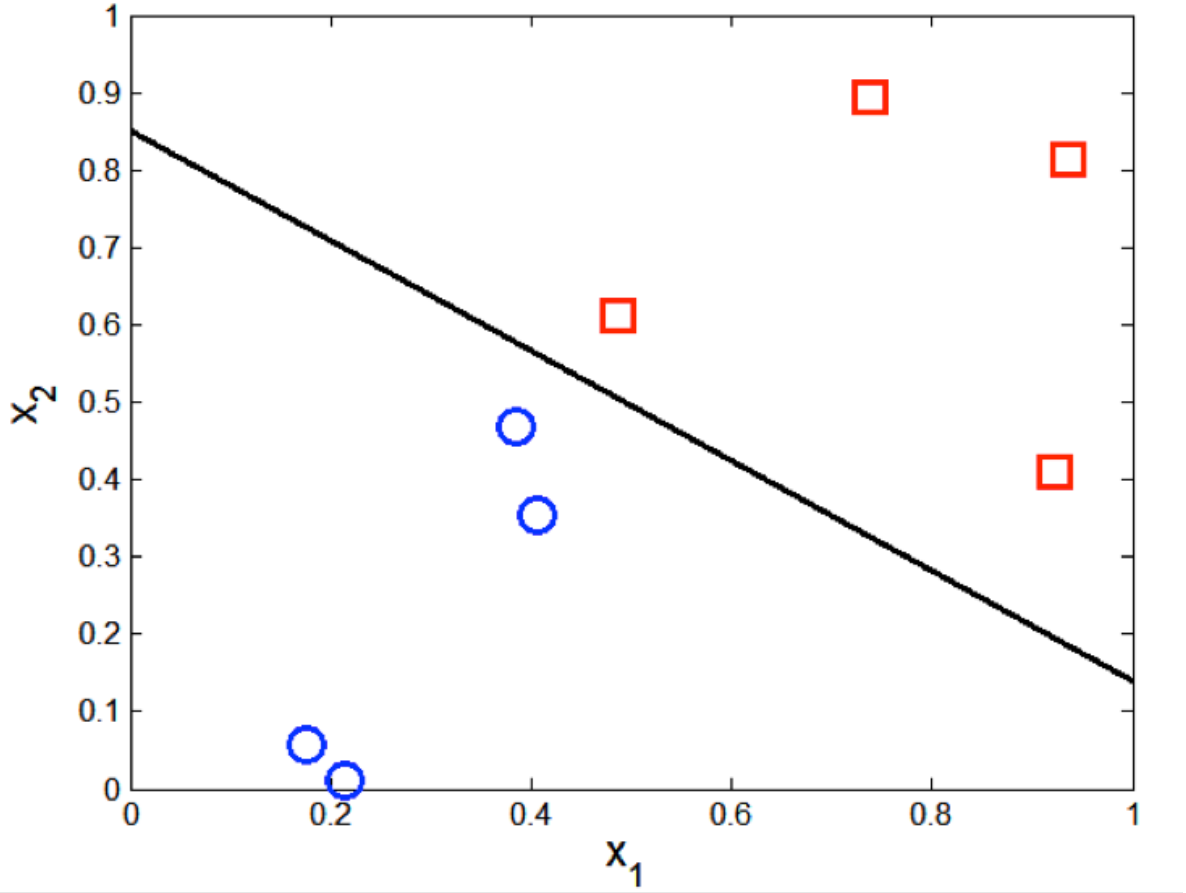
$b = \dfrac{b^{(1)} + b^{(2)}}{2} = \dfrac{-7.10604 - 6.97919}{2}$

$b = -7.0426$

Exemple (suite et fin)

- L'hyperplan solution est illustré dans la figure suivante

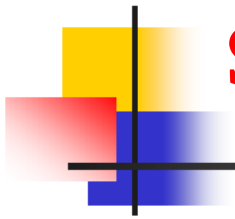
$$5.72609\, x_1 + 8.04366\, x_2 - 7.0426 = 0$$



Prédiction

- Comment classer les nouveaux objets?
- Une fois les paramètres de la solution sont obtenus, un nouvel objet **z** sera classé comme suit :

$$f(z) = sign(w^T \cdot z + b) = sign\left(\sum_{i=1}^{n} \lambda_i y_i x_i^T \cdot z + b\right)$$

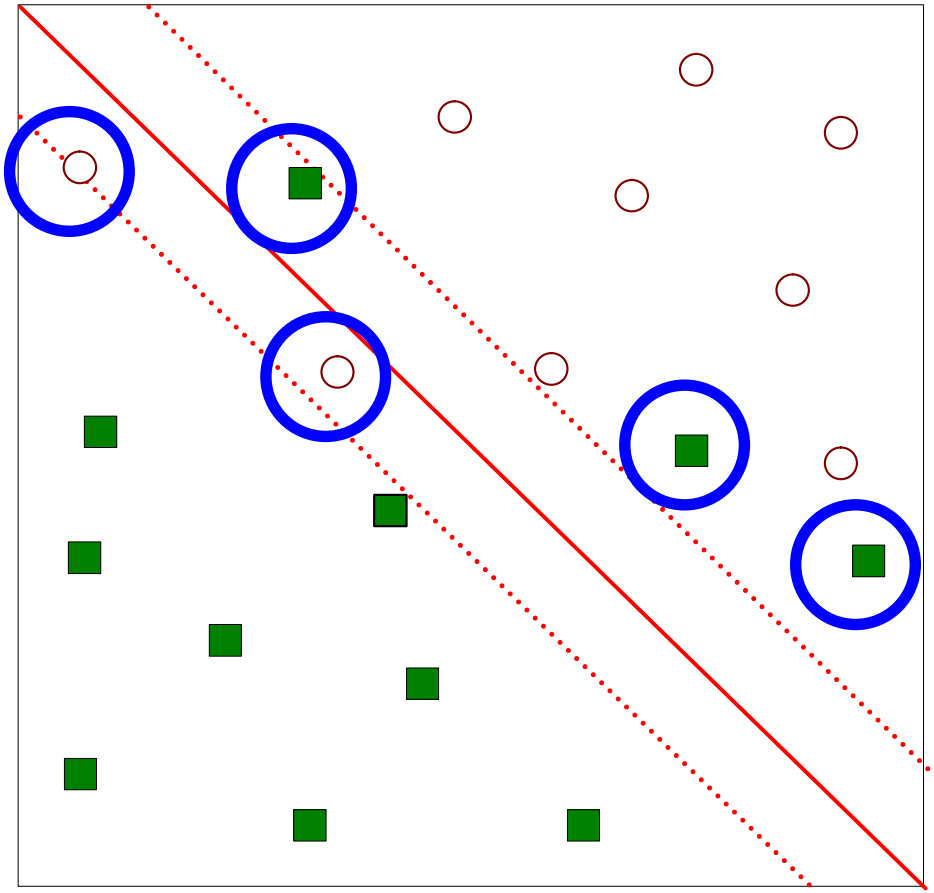


SVM linéaire pour les classes non séparables

Mise en contexte

- Jusqu'à maintenant, nous avons conservé l'hypothèse que les données sont linéairement séparables.
- Cependant, il est possible que les données d'entraînement comportent du bruit/données aberrantes (ex. erreur d'acquisition).
- Afin de ne pas apprendre le bruit et donc perdre une grande partie du pouvoir de généralisation, il est plus raisonnable d'admettre que certains exemples (supposés bruités) soient mal classés par notre classifieur.
- → SVM linéaire pour les classes non séparables réalise cette tâche.

Illustration



SVM linéaire pour les classes non séparables

- But : construire une fontainière de décision linéaire qui prend en compte le cas des classes non linéairement séparable. L'objectif est de trouver un bon équilibre entre conserver un chemin aussi large que possible et limiter les empiétements de marge (c.-à-d. le nombre d'observations se retrouvant à l'intérieur du chemin, voire même du mauvais côté). C'est ce qu'on appelle une classification à marge souple (en anglais, *soft margin* classification).

- Idée : relaxer les contraintes → utiliser une marge souple (*soft margin)* au lieu de la marge rigide (*hard margin)*.

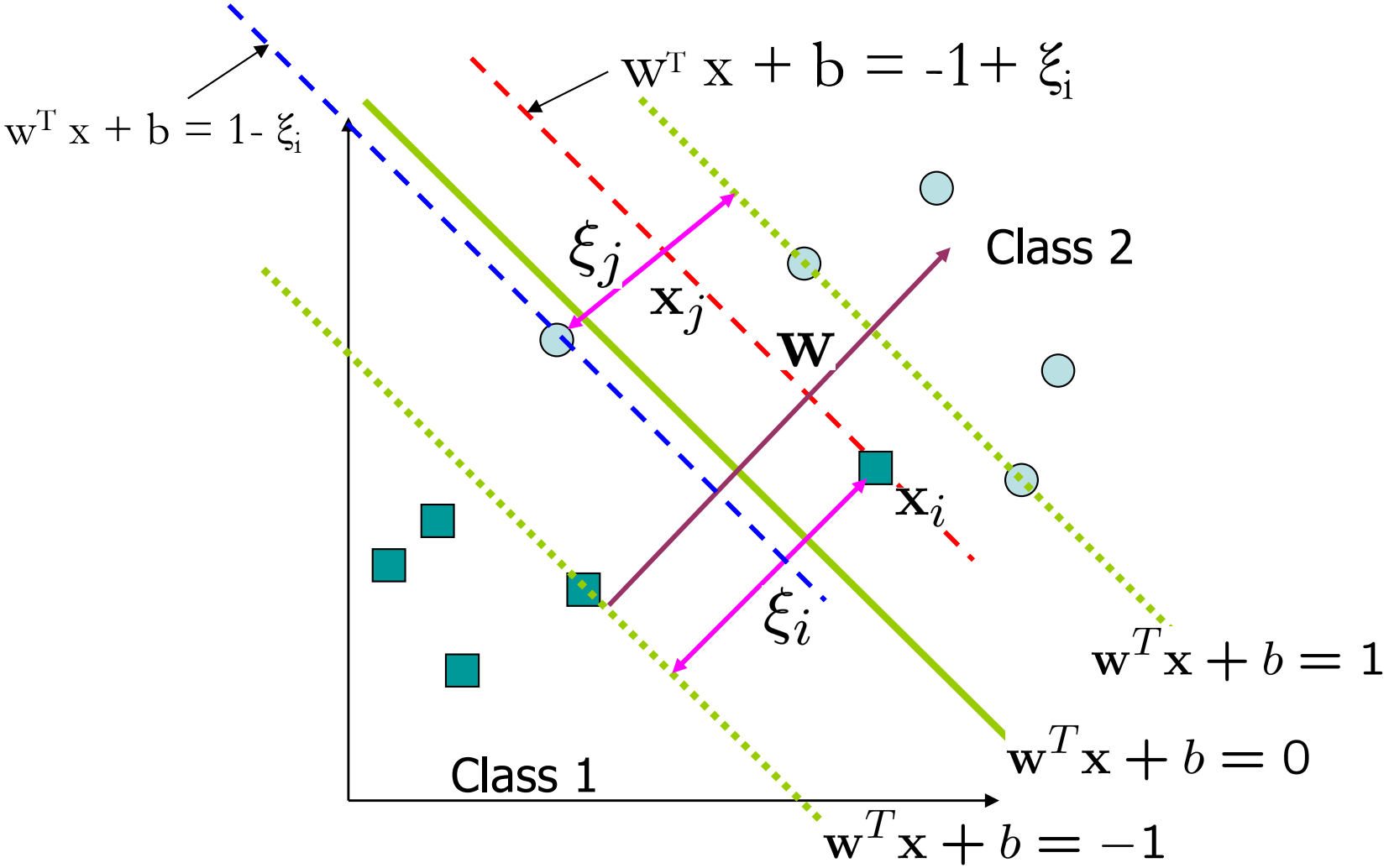
- Approche : formaliser et optimiser un compromis entre la largeur de la marge et le nombre d'exemples d'apprentissage mal classés.

Relaxer les contraintes

- Il faut relaxer les contraintes pour pouvoir accommoder les données non linéairement séparables.
- Cette tâche est accomplie par l'ajout des variables positives ξ nommées « slack variables » aux conditions du problème d'optimisation.
- Formellement

$$w^T \cdot x_i + b \geq 1 - \xi_i \;\; si \; y_i = 1,$$
$$w^T \cdot x_i + b \leq -1 + \xi_i \;\; si \; y_i = -1.$$
$$\forall i : \xi_i \geq 0$$

Illustration



ξ_i sont estimées à partir du résultat de la fonction $w^T . x + b$

Fonction objective du problème

La nouvelle fonction objective à minimiser est définie par

$$f(w) = \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{n} \xi_i\right) \qquad (7)$$

C : est un hyperparamètre qui règle le compromis entre w et ξ

L'équation (7) est optimisée sous les contraintes

$$w^T \cdot x_i + b \geq 1 - \xi_i \;\; si \; y_i = 1,$$

$$w^T \cdot x_i + b \leq -1 + \xi_i \;\; si \; y_i = -1.$$

Ces deux contraintes peuvent être réécrites comme

$$y_i(w^T \cdot x_i + b) \geq 1 - \xi_i$$

Formulation Lagrangienne du problème

Regrouper la fonction objective et les contraintes dans une seule expression

$$L_P = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i\left(y_i(w^T \cdot x_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n}\mu_i\xi_i \qquad (8)$$

Avec $||w||^2 = w^T w$

- Les deux premiers termes sont la fonction objective à minimiser.
- Le troisième terme représente les contraintes imposées au problème d'optimisation.
- Le dernier terme vient du fait qu'on impose que les variables ξ_i doivent être positives.

L'expression duale

Afin de minimiser le Lagrangien, on doit estimer les dérivées de L_p par rapport à w, b, ξ et puis les mètres à zéro :

$$L_P = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i\left(y_i(w^T \cdot x_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n}\mu_i\xi_i \qquad (8)$$

$$\frac{\partial L_P}{\partial w_i} = 0 \Rightarrow w_k = \sum_{i=1}^{n}\lambda_i y_i x_{ik}, \qquad (9)$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{n}\lambda_i y_i = 0. \qquad (10)$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow C - \lambda_i - \mu_i = 0 \Rightarrow \lambda_i + \mu_i = C \qquad (11)$$

L'expression duale

En substituant les équations (9), (10) et (11) dans (8) on trouve

$$L_P = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\lambda_i\left(y_i(w^T \cdot x_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n}\mu_i\xi_i$$

$$\Rightarrow \quad L_D = \frac{1}{2}\sum_{i=1}^{n}\lambda_i y_i x_i^T \sum_{j=1}^{n}\lambda_j y_j x_j + C\sum_{i=1}^{n}\xi_i$$

$$-\sum_{i=1}^{n}\lambda_i\left[y_i\left(\sum_{j=1}^{n}\lambda_j y_j x_j^T \cdot x_i + b\right) - 1 + \xi_i\right]$$

$$-\sum_{i=1}^{n}(C - \lambda_i)\xi_i$$

$$= \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j x_i^T \cdot x_j$$

Solution du problème d'optimisation

On obtient, comme dans le cas séparable, une formulation duale, excepté les contraintes qui sont légèrement différentes.

→ Le problème d'optimisation se réduit donc à trouver les multiplicateurs de Lagrange :

$$\max_{\lambda}\left\{\sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{n}\lambda_i\lambda_j y_i y_j (x_i^T \cdot x_j)\right\}$$

$$\forall i,\ j \quad 0 \le \lambda_i \le C$$

$$\sum_{i=1}^{n}\lambda_i y_i = 0$$

Solution du problème d'optimisation

L'estimation du paramètre w est identique au cas des classes séparables :

$$w_j = \sum_{i=1}^{n} \lambda_i y_i x_{ij}$$

- Le paramètre b est estimé à partir de

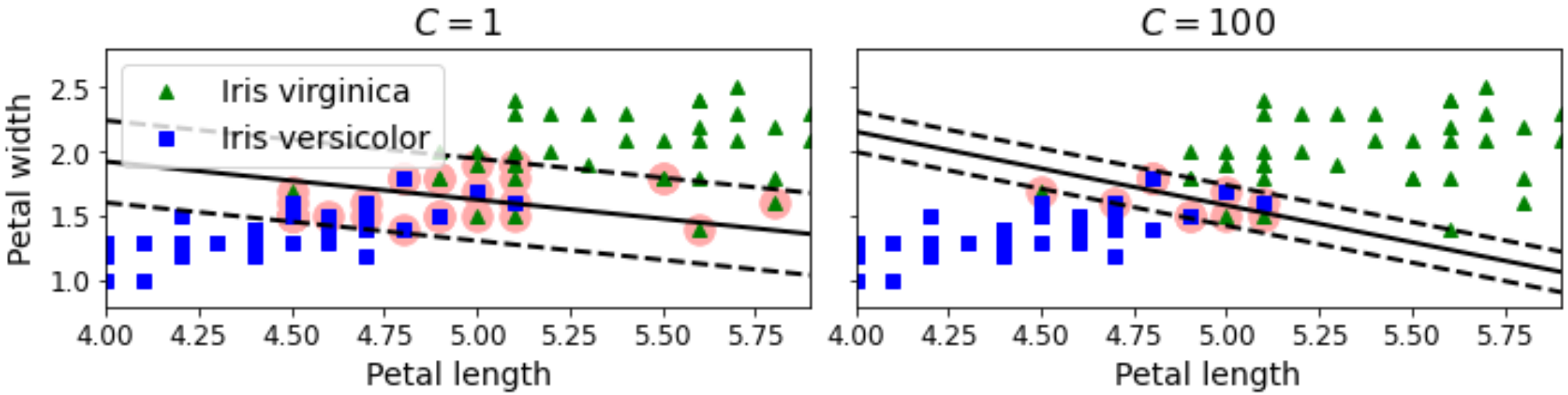
$$y_i(w^T \cdot x_i + b) - 1 + \xi_i = 0$$

Décision quant à la classification

Nous déterminant maintenant le statut d'un exemple x_i en regardant sa variable duale λ_i. Trois situations sont possibles :

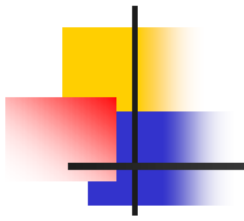
1. $\lambda_i = 0$: l'exemple est bien classé → l'exemple n'est pas un vecteur de support.
2. $0 < \lambda_i < C$: L'exemple est bien classé → c'est un vecteur de support.
3. $\lambda_i = C$: L'exemple est mal classé → il s'agit d'un bruit (l'exemple sera malgré tout considéré comme vecteur de support car $\lambda_i > 0$)

Exemple*



Si nous donnons à C une valeur basse, nous aboutissons au modèle de gauche. Avec une valeur élevée, nous obtenons le modèle de droite.
Il est généralement préférable d'avoir peu d'empiétements de marge.*

*Source : Aurélien Géron. Machine Learning avec Scikit-Learn : Mise en œuvre et cas concrets. Dunod, 2019.



SVM non linéaire

SVM non linéaire

- La formulation de SVM présentée jusqu'à maintenant permet de construire des surfaces séparatrices linéaires seulement.

→ Comment généraliser cette méthode pour les frontières de séparation non linéaire ?

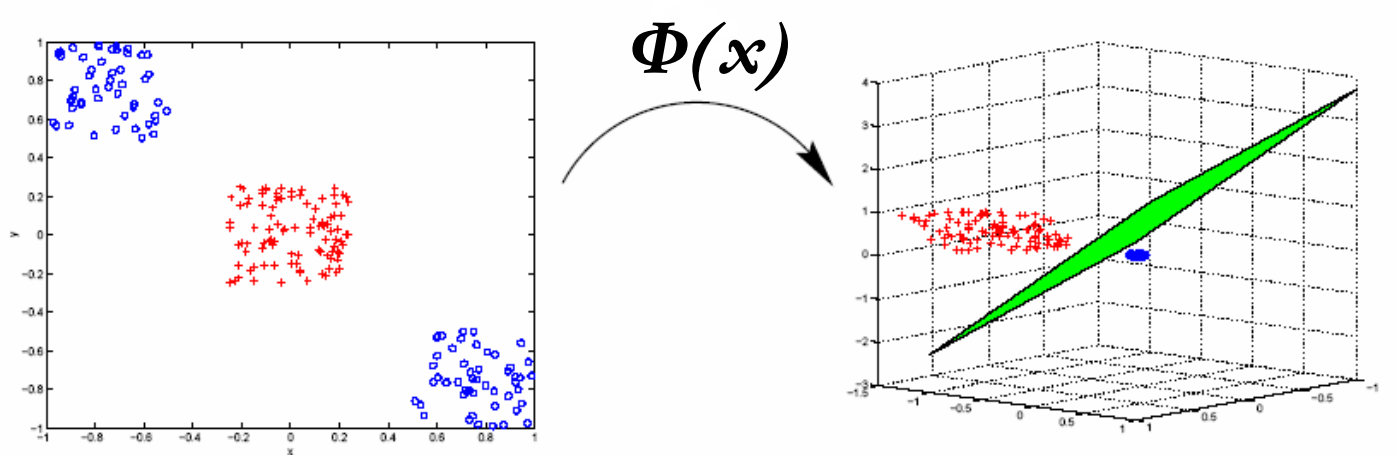
- L'idée consiste à transformer l'espace d'entrée X vers un espace de plus grande dimension afin de rendre le problème linéairement séparable.
- La transformation se fait à l'aide des fonctions de mapping $\Phi(X)$.
- Les données deviennent donc linéairement séparables et par conséquent la technique des SVM présentée précédemment peut être appliquée.

Passage dans un autre espace

- Plus la dimension de l'espace de description est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples et les contre-exemples est élevée.
- En transformant l'espace d'entrée en un espace de re-description de grande dimension, il devient donc possible d'envisager d'utiliser la méthode des SVM.
- Notons une transformation non linéaire de l'espace d'entrée X en un espace de redescription par $\Phi(X)$:

$$\boldsymbol{x} = (x_1, \ldots, x_d)^\top \quad \overset{\Phi}{\longmapsto} \quad \Phi(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_d(\boldsymbol{x}), \ldots)$$

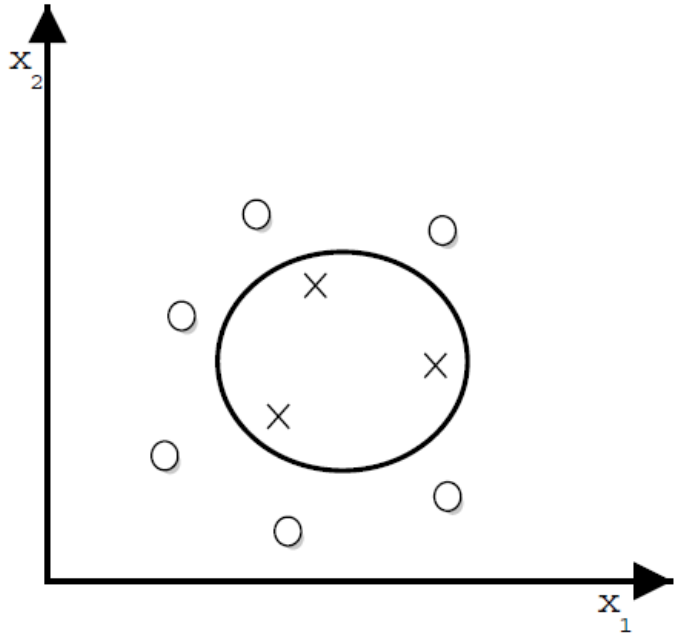
Illustration



(a) (b)

- (a) données originales.
- (b) données transformées.
- La dimension de l'espace des données transformées est généralement plus grande que de la dimension de l'espace des entrées .

Un autre exemple



- Les objets de l'ensemble de données illustré ci-dessous peuvent être classés selon la formulation suivante :

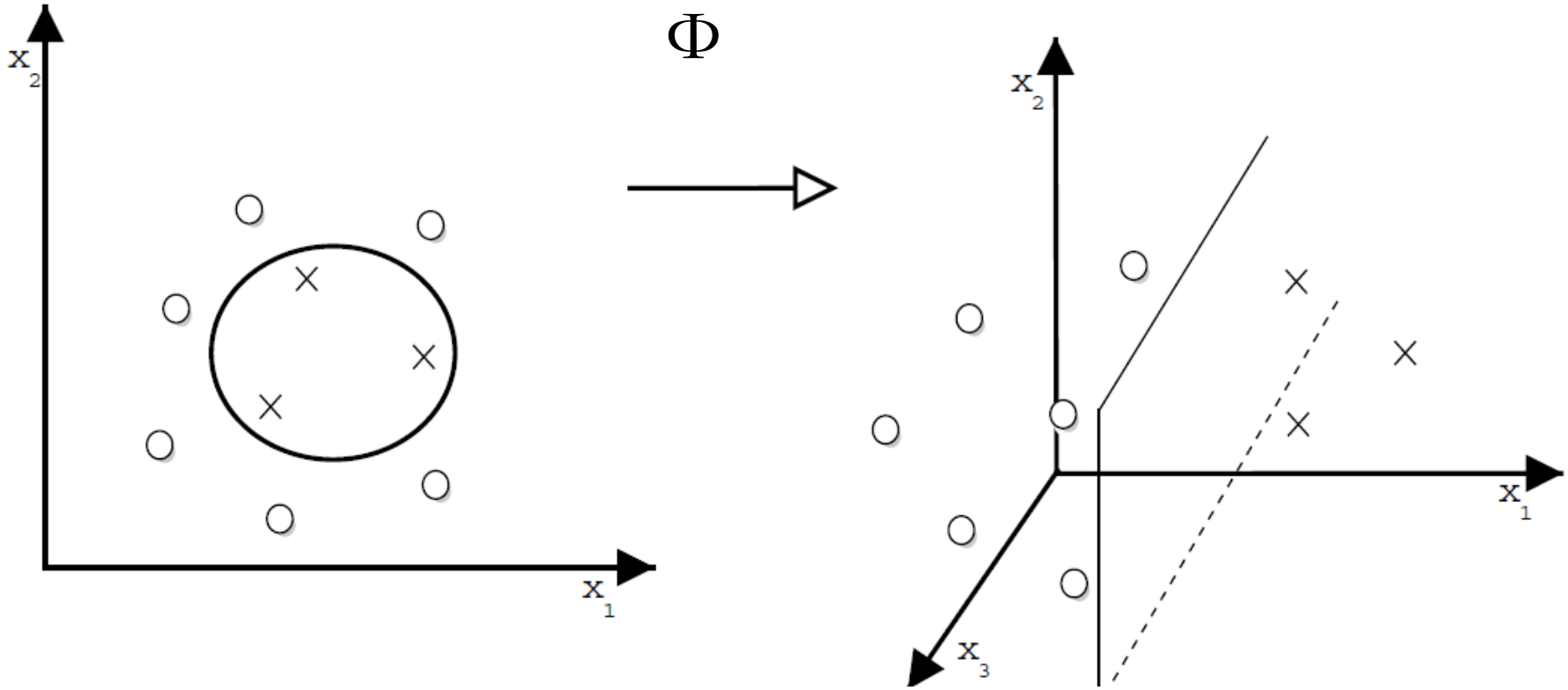
$$y(x_1, x_2) = \begin{cases} 1 & si \ \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & \text{sinon} \end{cases}$$

- L'équation de la surface séparatrice est donc :

$$\sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} = 0.2$$

Exemple (suite et fin)

- Illustration des données transformées



Un mapping Φ rendant les exemples *linéairement séparables*

Passage dans un autre espace

→ Le passage vers un nouvel espace implique l'augmentation de la dimension de l'ensemble de données d'apprentissage.

→ Le problème avec cette approche c'est qu'elle peut souffrir du problème de la malédiction de la dimensionnalité souvent présent dans les données de grande dimension.

→ On va illustrer que SVM évite ce problème.

Apprentissage d'un SVM non linéaire

- Soit Φ(x) une fonction qui permet de transformer les données d'apprentissage en un ensemble linéairement séparable dans un nouvel espace de plus grande dimension.
- Une fois la transformation est faite, on doit identifier une séparatrice linéaire pour pouvoir séparer les données.
- L'équation de la surface séparatrice linéaire de l'ensemble des données transformées est définie par :

$$\mathbf{w}^T \cdot \mathbf{\Phi(x)} + \mathbf{b} = 0$$

Apprentissage d'un SVM non linéaire

- La tâche d'apprentissage d'un SVM non linéaire peut être formalisée selon le problème d'optimisation suivant :

$$\min_{w} = \frac{\|w\|^2}{2}$$

avec la contrainte $y_i(w^T \cdot \Phi(x_i) + b) \geq 1, \ i = 1, 2, \cdots, n.$

→ Cette formulation est identique à celle du SVM linéaire.

→ La différence majeure, c'est qu'au lieu de travailler avec les attributs originaux **x**, la méthode propose de focaliser plutôt sur les attributs transformés **Φ(x)**

Formulation duale du problème

En utilisant la même stratégie adoptée pour les SVM linéaires, le problème d'optimisation se transcrit par

$$\underset{\lambda}{Max}\left\{\sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{n}\lambda_i\lambda_j y_i y_j \Phi(x_i)^T \cdot \Phi(x_j)\right\}$$

$$\lambda_i \geq 0, i = 1, \cdots, n$$

$$\sum_{i=1}^{n}\lambda_i y_i = 0$$

→ Les valeurs λ_i peuvent être identifiées par l'utilisation des techniques de programmation quadratique.

Solution au problème de l'optimisation

Les valeurs de w et *b* peuvent être identifiées suivant les deux équations suivantes :

$$w = \sum_{i=1}^{n} \lambda_i y_i \Phi(x_i)$$

$$\lambda_i \left[y_i \left(\sum_{j=1}^{n} \lambda_j y_j \Phi(x_j)^T \cdot \Phi(x_i) + b \right) - 1 \right] = 0$$

$\Phi(x_j)^T$. $\Phi(x_i)$ dénote le produit scalaire dans le nouvel espace.

Prédiction

- Comment classer les nouveaux objets?
- Une fois les paramètres de la solution sont obtenus, un nouvel objet z sera classé comme suit :

$$f(z) = sign(w^T \cdot \Phi(z) + b) = sign\left(\sum_{i=1}^{n} \lambda_i y_i \Phi(x_i)^T \cdot \Phi(z) + b\right)$$

Remarque

- $\Phi(x_j)^T \cdot \Phi(x_i)$ devient rapidement très couteux à calculer quand la dimension de X augmente, ceci d'autant plus que l'on utilisera des transformations non linéaires des descripteurs d'entrée.

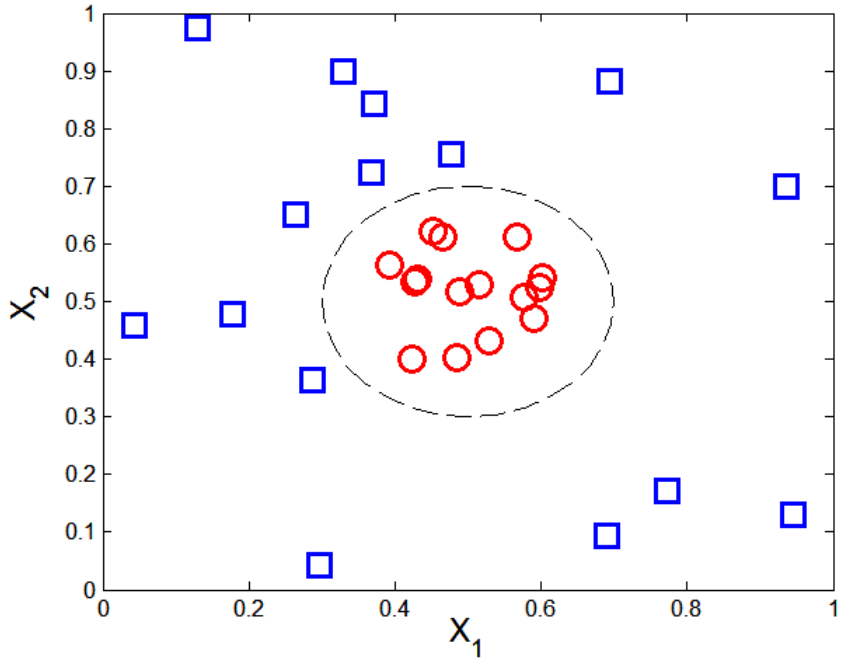
- Pour pallier ce problème, il existe des fonctions bilinéaires symétriques positives **$K(x_j, x_i)$,** appelées **fonctions noyau**, faciles à calculer, et dont on peut montrer qu'elles correspondent à un produit scalaire $\Phi(x_j)^T \cdot \Phi(x_i)$ dans un espace de grande dimension.

Les fonctions noyaux

- Le produit scalaire est souvent utilisé pour mesurer la similarité entre deux vecteurs (ex. distance de cosine).
- De même, le produit scalaire $\Phi(x_j)^T \cdot \Phi(x_i)$ peut être aussi considéré comme une mesure de similarité entre deux objets x_i et x_j dans l'espace transformé.
- En utilisant le mapping Φ, on peut définir une mesure de similarité dans l'espace transformé: $K(x_i, x_j) = \Phi(x_j)^T \cdot \Phi(x_i)$. La fonction K(., .) est appelée noyau.
- Pour calculer l'hyper plan optimal dans l'espace transformé, il suffit de remplacer toutes les occurrences du produit scalaire par la fonction noyau.

Les fonctions noyaux

Soit l'ensemble de données suivant :



On veut réaliser une transformation vers un espace de plus grande dimension en utilisant un mapping Φ suivant

$$\Phi : (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

Les fonctions noyaux

Le produit scalaire entre deux vecteurs se calcule comme suit :

$$\Phi(u)^T \cdot \Phi(v) = \Phi\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right)^T \cdot \Phi\left(\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}\right)$$

$$= (u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1) \cdot \begin{pmatrix} v_1^2 \\ v_2^2 \\ \sqrt{2}v_1 \\ \sqrt{2}v_2 \\ 1 \end{pmatrix}$$

$$= u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 + 1$$

$$= (u^T \cdot v + 1)^2$$

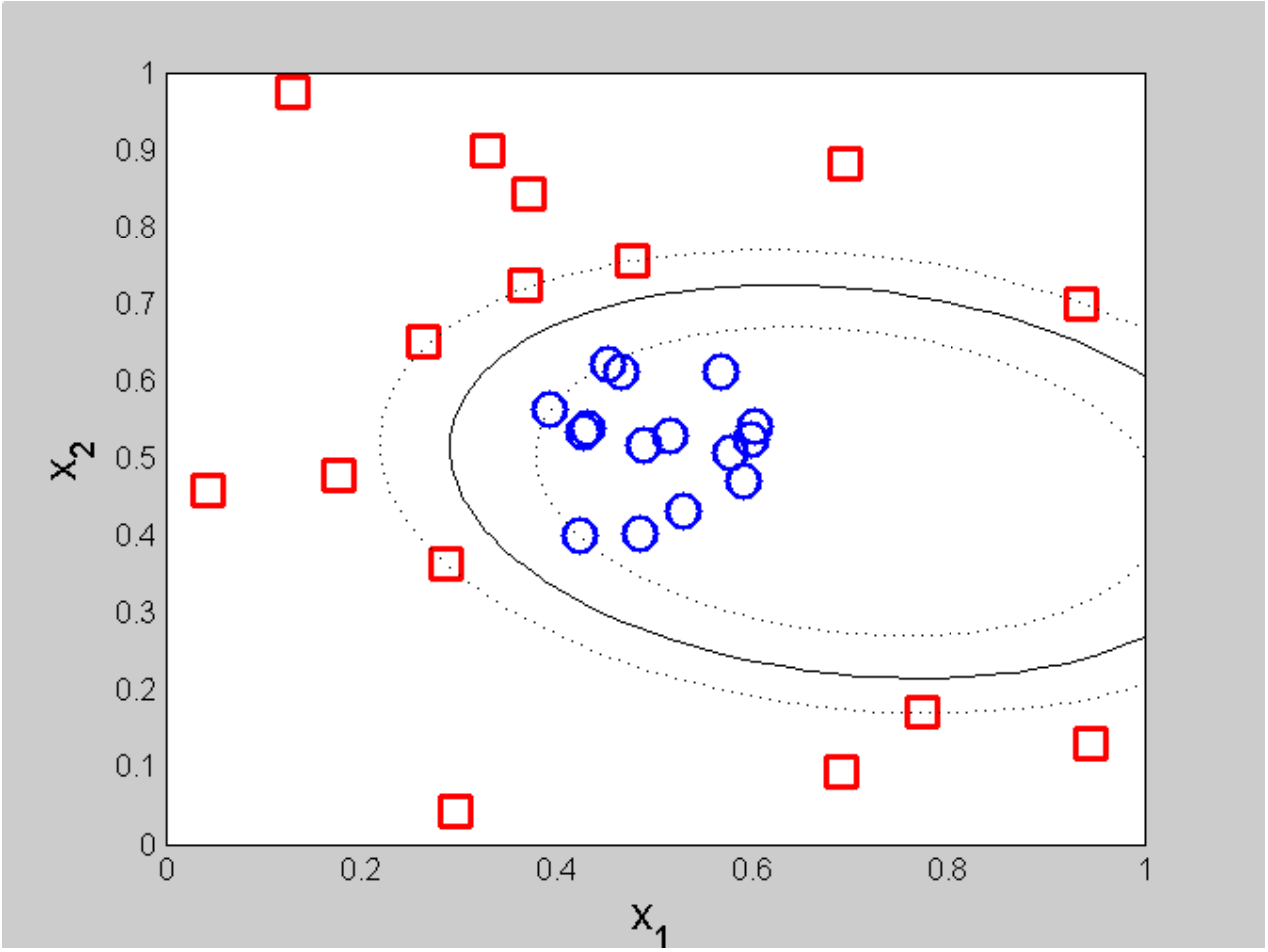
Les fonctions noyaux

→ Le développement illustré précédemment montre que le produit scalaire dans l'espace transformé peut être exprimé comme une mesure de similarité dans l'espace original.

$$\mathbf{K(u, v) = \Phi(u)^T . \Phi(v) = (u^T . v + 1)^2}$$

→ On retrouve une fonction noyau polynomiale.

Exemple



Cette figure illustre la surface séparatrice identifier par SVM en utilisant la fonction noyau polynomiale $K(u, v) = (u^T . v + 1)^2$

Classer un nouvel objet

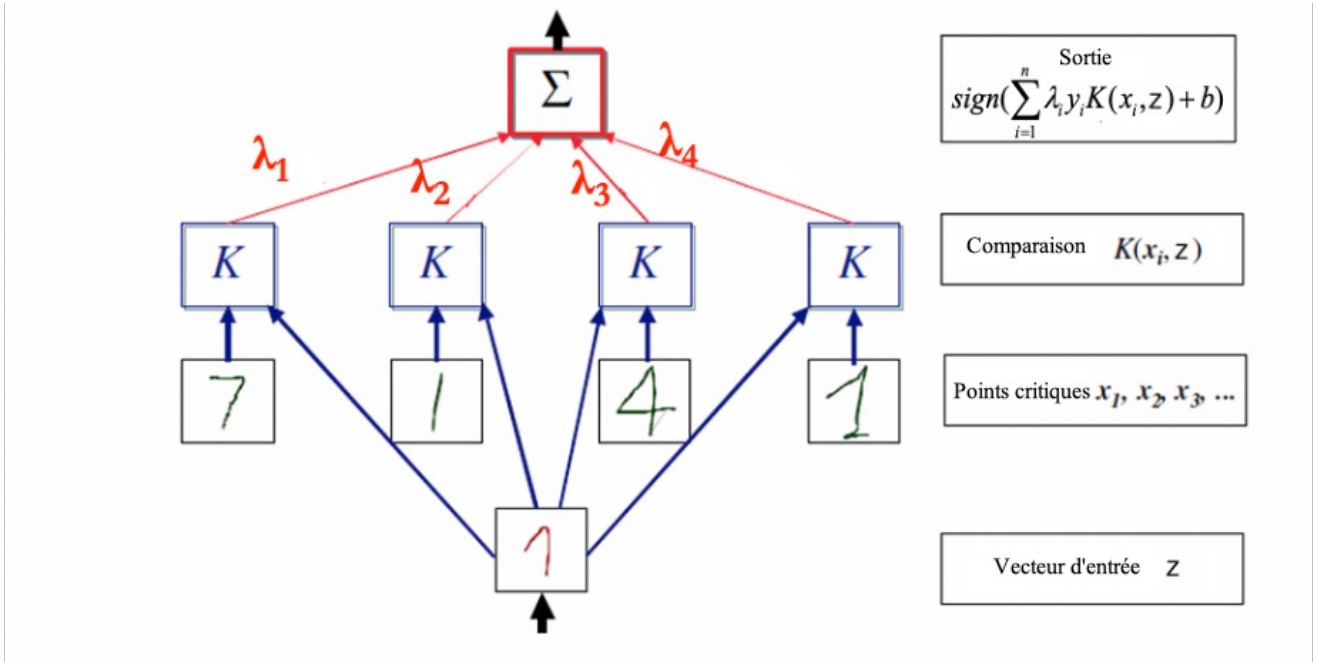
Un nouvel objet est classé selon les équations suivantes :

$$f(z) = sign\left(\sum_{i=1}^{n} \lambda_i y_i \Phi(x_i)^T \cdot \Phi(z) + b \right)$$

$$= sign\left(\sum_{i=1}^{n} \lambda_i y_i K(x_i, z) + b \right)$$

$$= sign\left(\sum_{i=1}^{n} \lambda_i y_i (x_i^T \cdot z + 1)^2 + b \right)$$

Classer un nouvel objet



Cette figure résume le fonctionnement des séparateurs à vastes marges et montre le rôle des fonctions noyaux. Lors de l'apprentissage, ici de chiffres manuscrits, un certain nombre d'exemples critiques sont retenus pour définir la fonction de décision (ici deux exemples positifs de '1' et deux exemples négatifs, un '4' et un '7'). Lorsqu'une nouvelle entrée est présentée au système, elle est comparée aux exemples critiques à l'aide des fonctions noyaux qui réalisent un produit scalaire dans l'espace de redescription $\Phi(X)$. La sortie est calculée en faisant une somme pondérée (une combinaison linéaire) de ces comparaisons.

Source : A. Cornuéjols, L. Miclet, V. Barra, « Apprentissage artificiel: 3e édition ». Eyrolles, 2018.

Avantages des fonctions noyaux

→ Suite à l'utilisation des fonctions noyaux, on n'a pas besoin de savoir la forme exacte de la fonction de mapping Φ. Cependant, les fonctions noyaux utilisées doivent satisfaire des conditions bien précises appelées condition de Mercer.

→ Le calcul du produit scalaire en utilisant les fonctions noyaux est nettement moins couteux que le produit scalaire $\Phi(x_j)^T \cdot \Phi(x_i)$

→ Puisque le calcul avec les fonctions noyaux est effectué dans l'espace de donnée originale, les problèmes liés à la malédiction de la dimension peuvent être évités.

Condition de Mercer

Avant d'énoncer le théorème de Mercer, voici quelques définitions utiles.

- **Définition 1 (matrice de Gram)**

La matrice contenant les similarités entre tous les exemples d'apprentissage G est appelée matrice de Gram

$$G = \begin{pmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{21} & k_{22} & \cdots & k_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ k_{n1} & k_{n2} & \cdots & k_{nn} \end{pmatrix}$$

Condition de Mercer

- **Définition 2 (matrice définie positive)**

Une matrice *M* de taille *n* x *n*, dont les éléments sont des réels, est définie positive SSI toutes les valeurs propres de M soient positives.

- **Théorème (condition de Mercer)**

La fonction K(., .) est une fonction noyau SSI

$$G = \left(K(x_i, x_j)\right)_{i,j=1}^{n}$$

est définie positive

Noyaux courants

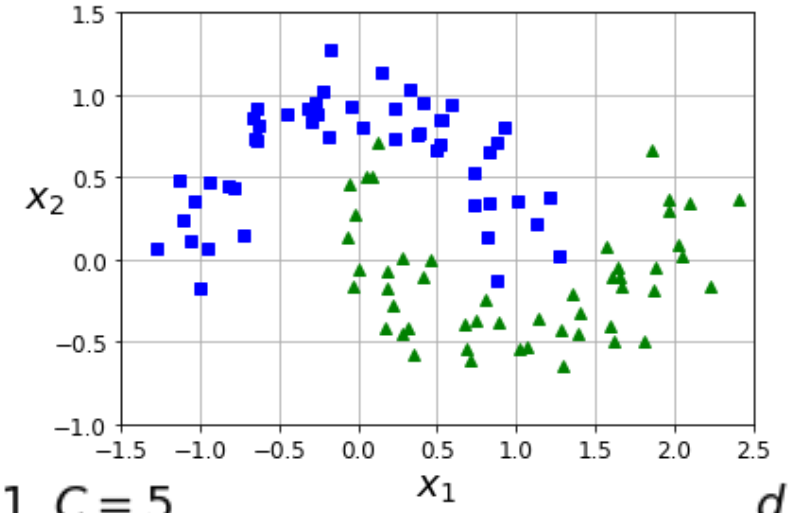
$$\text{Linéaire :} \quad K(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^T \boldsymbol{v}$$

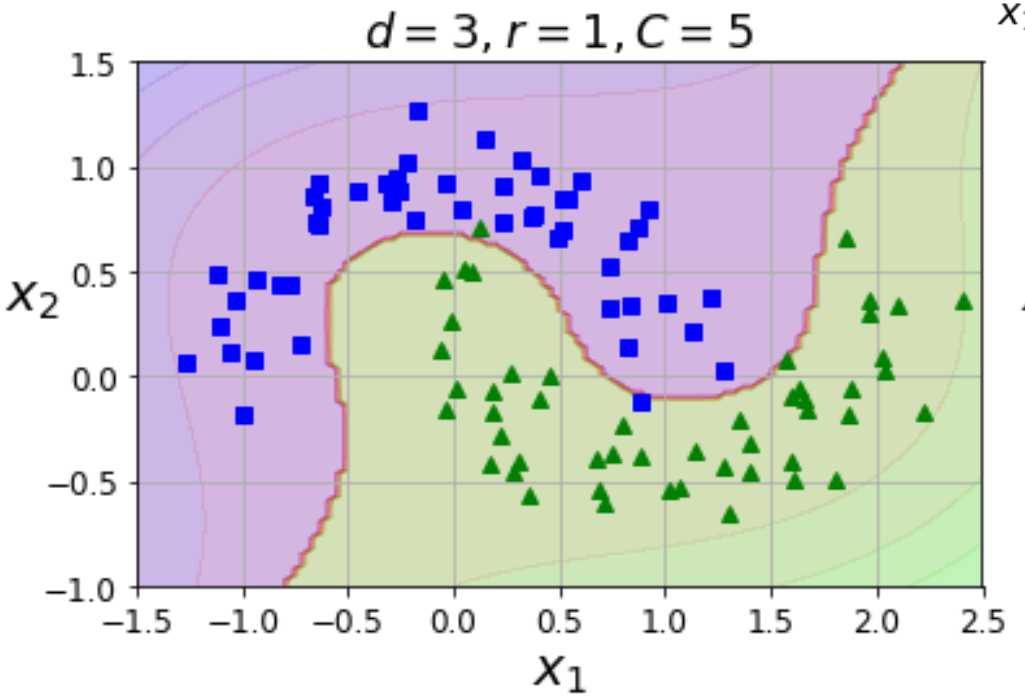
$$\text{Polynomial :} \quad K(\boldsymbol{u}, \boldsymbol{v}) = \left(\gamma\, \boldsymbol{u}^T \boldsymbol{v} + r\right)^d$$

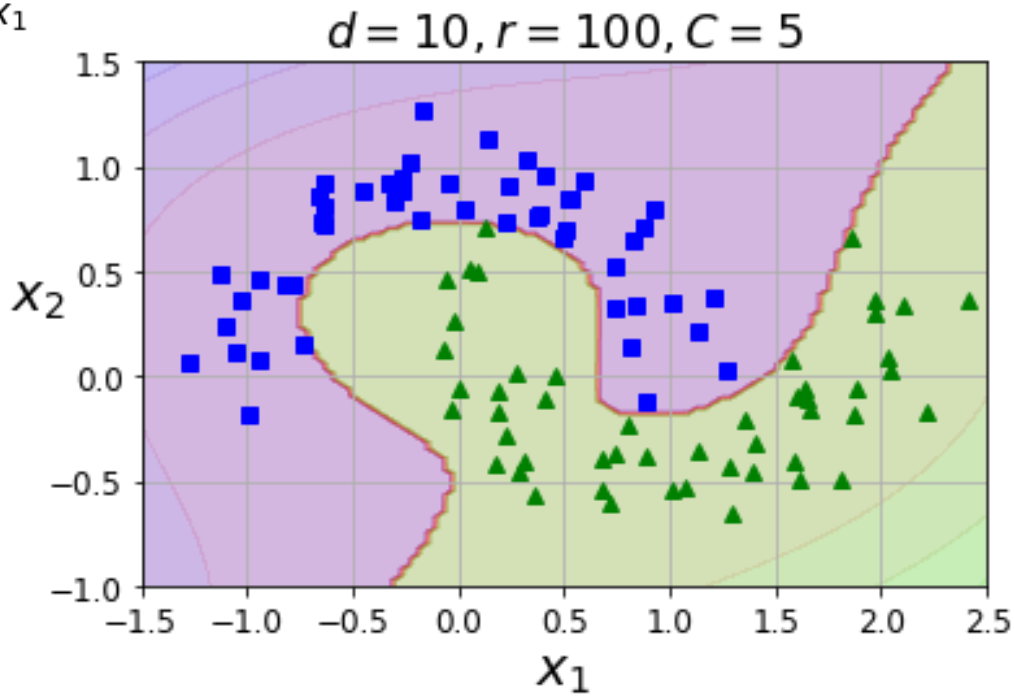
$$\text{Radial gaussien :} \quad K(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\gamma \left\| \boldsymbol{u} - \boldsymbol{v} \right\|^2\right)$$

$$\text{Sigmoïde :} \quad K(\boldsymbol{u}, \boldsymbol{v}) = \tanh\left(\gamma\, \boldsymbol{u}^T \boldsymbol{v} + r\right)$$

Exemple avec un noyau polynomial*

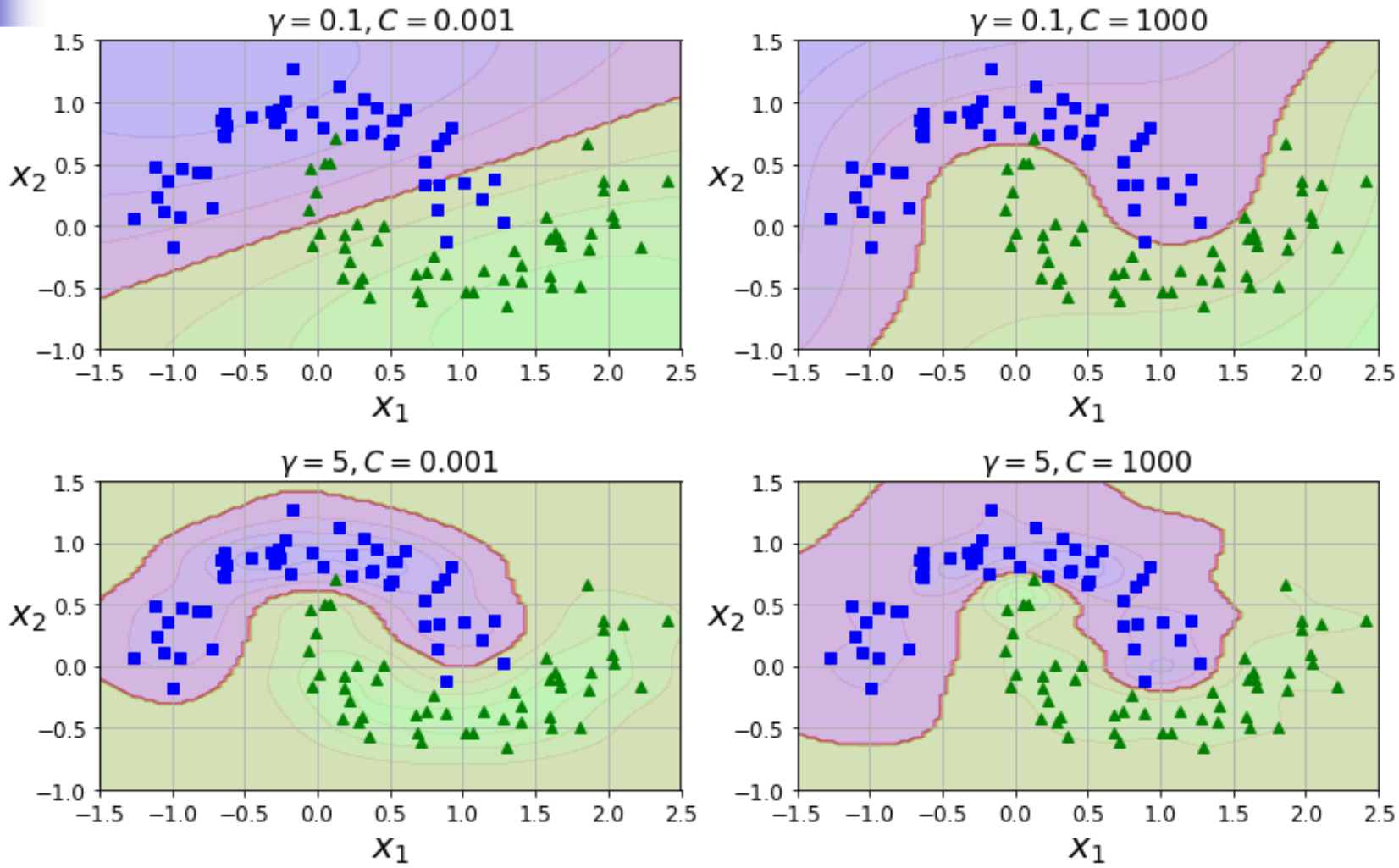






*Source : Aurélien Géron. Machine Learning avec Scikit-Learn : Mise en œuvre et cas concrets. Dunod, 2019.

Exemple avec un noyau radial gaussien*



γ fonctionne comme un hyperparamètre de régularisation : si votre modèle surajuste, vous devez le réduire ; et s'il sous-ajuste, vous devez l'augmenter (de manière similaire à l'hyperparamètre C).

*Source : Aurélien Géron. Machine Learning avec Scikit-Learn : Mise en œuvre et cas concrets. Dunod, 2019.

La mise en œuvre des SVM

- La mise en œuvre de la méthode des séparateurs à vaste marge (SVM) requiert l'accès à un système de résolution de programmation quadratique calculant les variables duales λ_i (les Lagrangiens).

  → Le code de quelques systèmes de résolution de programmation quadratique est disponible à partir du lien suivant : **http://www.numerical.rl.ac.uk/qp/qp.html**

- Le choix des autres paramètres à savoir la fonction noyau et de la valeur de *C* sont souvent faits en utilisant une méthode de validation croisée en testant l'effet de différents choix.

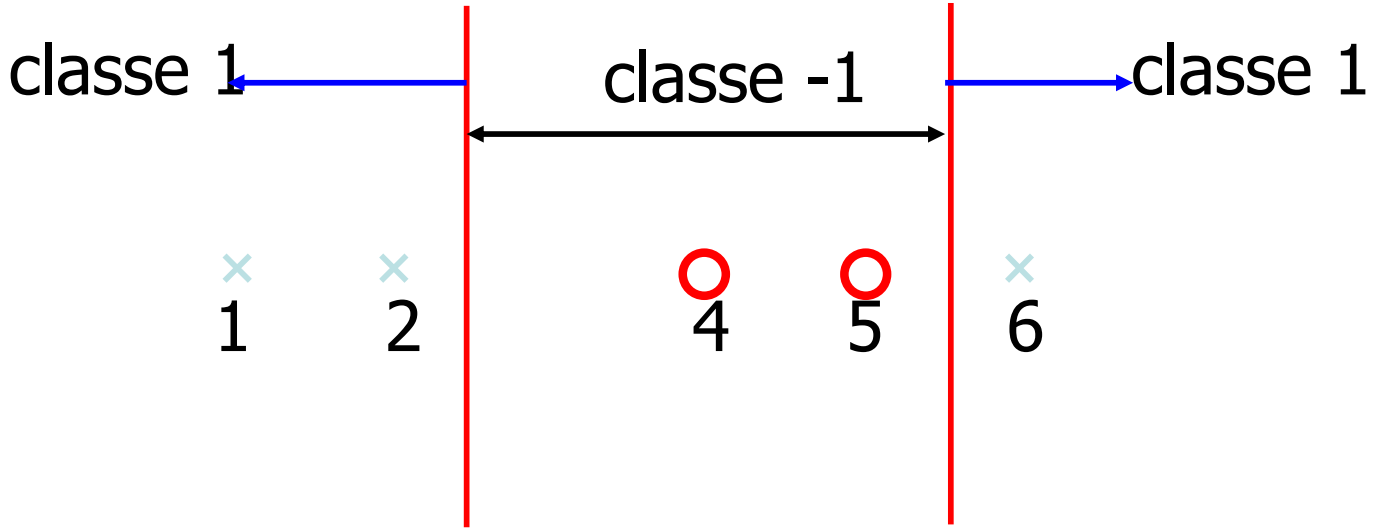
Exemple : SVM non linéaire*

Utiliser une fonction polynomiale de degré 2 :

$$K(u,v) = (u^T . v + 1)^2$$

Pour identifier la surface séparatrice pour l'ensemble suivant

Objets	$x_1=1$	$x_2=2$	$x_3= 4$	$x_4=5$	$x_5=6$
Classe	1	1	-1	-1	1



*Source : A. Cornuéjols, L. Miclet, V. Barra, « Apprentissage artificiel: 3e édition ». Eyrolles, 2018.

Exemple (suite)

Premièrement on doit identifier les coefficients de Lagrange

$$\underset{\lambda}{Max}\left\{\sum_{i=1}^{5}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{5}\lambda_i\lambda_j y_i y_j \Phi(x_i)^T \cdot \Phi(x_j)\right\}$$

$$= \underset{\lambda}{Max}\left\{\sum_{i=1}^{5}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{5}\lambda_i\lambda_j y_i y_j K(x_i, x_j)\right\}$$

$$= \underset{\lambda}{Max}\left\{\sum_{i=1}^{5}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{5}\lambda_i\lambda_j y_i y_j (x_i^T \cdot x_j + 1)^2\right\}$$

En utilisant les techniques de programmation quadratique, on trouve : $\lambda_1 = 0, \lambda_2 = 2.5, \lambda_3 = 0, \lambda_4 = 7.333, \lambda_5 = 4.833$

Les vecteurs de support sont donc: $\{x_2=2, x_4=5, x_5=6\}$

Source : A. Cornuéjols, L. Miclet, V. Barra, « Apprentissage artificiel: 3e édition ». Eyrolles, 2018.

Exemple (suite)

- L'équation de la fonction discriminante est définie par

λ_5 y_5 $K(z, x_5)$

$$f(z) = \sum_{i=1}^{n} \lambda_i y_i K(x_i, z) + b$$

$$= 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + 4.833(1)(6z+1)^2 + b$$

$$= 0.6667z^2 - 5.333z + b$$

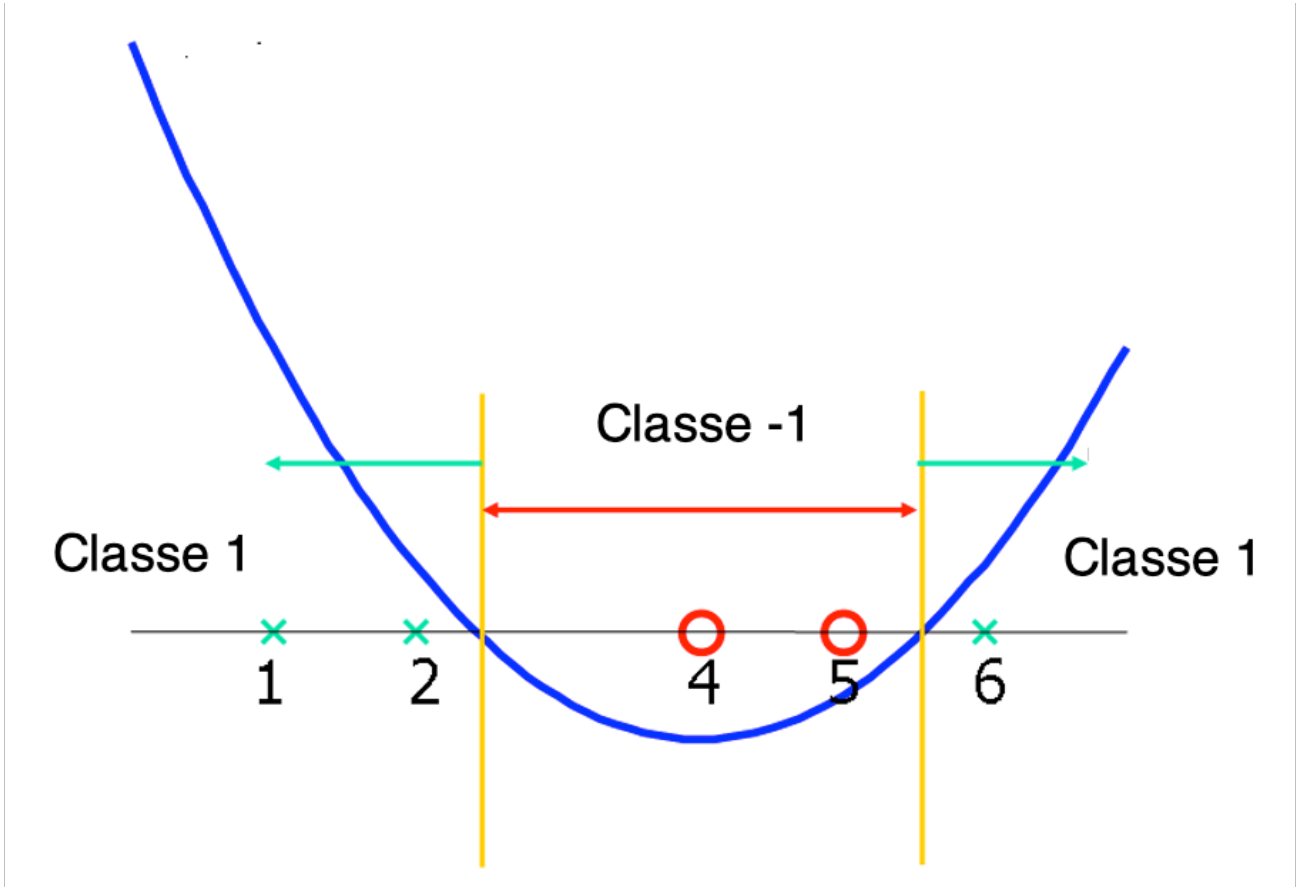
- b peut être identifié en considérant seulement les vecteurs de support : c.-à-d. résoudre f(2)=1 et f(5) = -1 et f(6) = 1 et puis prendre la valeur moyenne → $b = 9$

→ L'équation de la surface séparatrice est donc :

$$f(z) = 0.6667z^2 - 5.333z + 9$$

Source : A. Cornuéjols, L. Miclet, V. Barra, « Apprentissage artificiel: 3e édition ». Eyrolles, 2018.

Exemple (suite et fin)



Source : A. Cornuéjols, L. Miclet, V. Barra, « Apprentissage artificiel: 3e édition ». Eyrolles, 2018.

Exemple de code

Exemples d'implémentation avec Scikit-leran

https://scikit-learn.org/stable/auto_examples/index.html#support-vector-machines

Les problèmes multiclasse

- SVM peut être bien adapté aux problèmes multiclasse.
- L'idée consiste à utiliser des méthodes de décomposition permettent d'aborder un problème de discrimination à classes multiples comme une combinaison de problèmes de classification binaire.

Approche « un contre tous »

- $Y = \{y_1, y_2, \ldots, y_k\}$ désigne un ensemble de *k* classes
- L'approche consiste à décomposer le problème multiclasses en un problème de *k* classificateur binaire → utiliser un classificateur binaire par classe.
- Pour chaque classe y_i (i = 1, …, k), un problème binaire est créé de telle sorte que les objets de la classe y_i sont considérés comme positifs, alors que les objets restants sont considérés comme négatifs.
- À partir de cette décomposition, un classificateur binaire est défini afin de séparer les objets de la classe y_i du reste.
- Pour affecter un nouvel exemple, on le présente donc à *k* classificateurs, et la décision s'obtient en application du principe "winner-takes-all"

Approche « un contre un »

- Une approche qui consiste à utiliser un classificateur par couple de classes (y_i, y_j). Donc il faut construire *k(k-1)/2* classificateurs binaires.
- Lors de la construction d'un classificateur pour distingué y_i de y_j les objets qui ne font pas partis des deux classes y_i et y_j sont ignorés.
- Le classificateur indicé par le couple (i, j), est destiné à distinguer la classe d'indice i de celle d'indice j.
- Pour affecter un exemple, on le présente donc à *k(k-1)/2* classificateurs binaire, et la décision s'obtient habituellement en effectuant un vote majoritaire.

DAGSVM*

- DAGSVM est un modèle de discrimination multiclasses dont l'architecture est un graphe de décision orienté sans cycle avec pour étiquettes de ses nœuds des SVM (à deux classes).

- Un nœud donné est associé à une liste de classes auxquelles l'exemple d'intérêt peut appartenir.

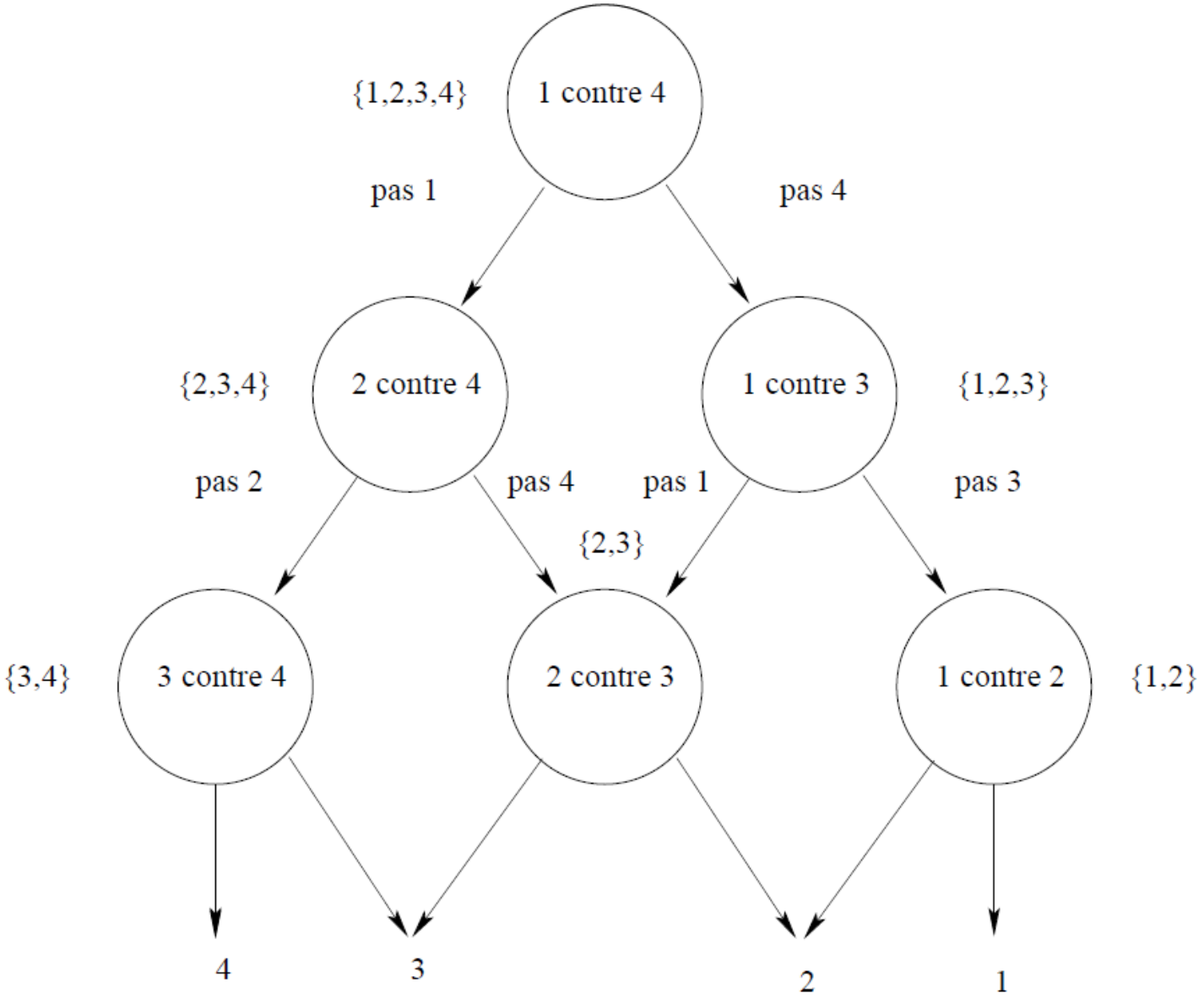
* J.C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classication. NIPS, pages 547 - 553, 2000.

DAGSVM

- SVM effectue une décision entre les deux classes aux extrémités de la liste : les classes 1 et k pour le SVM situé à la racine, 2 et k pour le SVM situé sur le fils gauche de la racine, 1 et $k-1$ pour le SVM situé sur le fils droit de la racine et ainsi de suite.
- Les nœuds de la couche d'indice $k-1$ produisent une décision en séparant les deux seules catégories contenues dans leur liste.

DAGSVM : illustration

DAGSVM pour un problème de quatre classes



Méthode des codes correcteurs d'erreurs

- Le principe consiste à représenter les catégories par des mots binaires différents, mais de même taille.
- En notant N la taille de ces mots, on obtient ainsi une matrice M avec k lignes (k le nombre de classe) et N le nombre de colonnes.
- Les colonnes représentent donc un partitionnement en deux superclasses l'ensemble des k classes.
- Pour chaque colonne (partition), on va entraîner un classificateur binaire.
- Ainsi, chaque exemple est associé à un vecteur de $\{0, 1\}^N$. Il est affecté à la classe correspondant au mot code le plus proche de ce vecteur au sens de la distance de Hamming.

Exemple

- On suppose que nous avons un problème de 4 classes.
- Le tableau suivant illustre les mots codes associés à chaque classe (ici on suppose que $N=7$)

Classe	Mots codes						
y1	1	1	1	1	1	1	1
y2	0	0	0	0	1	1	1
y3	0	0	1	1	0	0	1
y4	0	1	0	1	0	1	0

- Chaque colonne désigne un partitionnement en deux superclasses.
- Un classificateur binaire est entraîné pour chaque ensemble de données défini par chaque colonne.

Exemple (suite)

Pour classer un nouvel objet : rouler les 7 classificateurs entraînés précédemment pour prédire la classe de l'objet en question.

- Donc pour un seul objet on va lui associer un mot binaire de taille 7 qui contient les décisions obtenues par les 7 classificateurs.
- Par exemple (0, 1, 1, 1, 1, 1, 1)

→Utiliser la distance de Hamming pour trouver le mot binaire le plus proche à partir de la table des mots de codes.

→ Expl: distance de Hamming entre (1,1,1,1,1,1,1) et (0,1,1,1,1,1,1) = 1

Exemple (suite et fin)

→Distance de Hamming entre (0,1,1,1,1,1,1) et (0,0,0,0,1,1,1) est 3 …

Puisque la distance entre le mot code de la classe y_1 (c.-à-d. 1,1,1,1,1,1,1) et mot code de l'objet à classer (c.-à-d. 0,1,1,1,1,1,1) est la plus petite donc l'objet à classer appartient à la classe y_1.

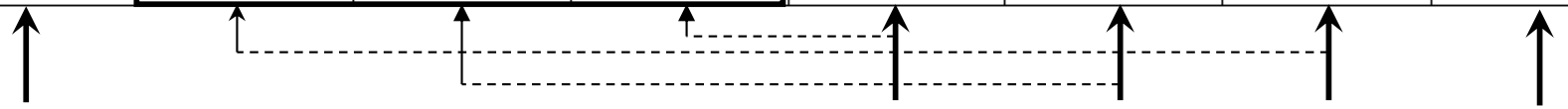
Méthode des codes correcteurs d'erreurs

- Cette approche est d'autant plus efficace que les mots codes son plus distant les uns des autres (toujours au sens de la distance de Hamming).
- En fait, il convient de maximiser non seulement la séparation des lignes de la matrice, mais aussi la séparation de ses colonnes. Si cette dernière séparation n'est pas assurée, alors les classificateurs associés à des colonnes proches risquent d'effectuer des erreurs similaires (corrélées).
- Or, l'utilisation de cette approche n'est efficace que si les erreurs effectuées sur les différentes colonnes sont relativement peu corrélées.

Méthode des codes correcteurs d'erreurs

- Toutes les colonnes possibles pour un problème de trois

Classe	Mots codes							
	F1	F2	F3	F4	F5	F6	F7	F8
y1	0	0	0	0	1	1	1	1
y2	0	0	1	1	0	0	1	1
y3	0	1	0	1	0	1	0	1



→ Il faut éliminer toutes les colonnes qui contiennent que des 0 ou des 1. Il faut aussi éliminer les colonnes qui sont la négation d'autres colonnes.

→ Donc pour un problème de *k* classes, il y aura au plus $2^{k-1}-1$ colonnes utilisables.