Support Vector Machines (SVM)

Plan

- Introduction
- Classification binaire –Séparation linéaire
- 2.Maximisation de la marge (I) –Formulation initiale
- 3.Maximisation de la marge (II) –Formulation duale
- 4.Cas de la séparation imparfaite –Soft Margin
- 5.Fonction noyau –Séparation non linéaire
- 8.Extension aux problèmes multi classes
- 10.Bilan –Avantages et inconvénients

SVM : **Introduction**

- ✓ Les machines à vecteurs de support ont été inventées par V.Vapnik et ses collègues dans les années 1970 en Russie et est devenu connu de l'Occident en 1992,
- ✓ Les SVM sont des classifieurs linéaires qui consistent à trouver un hyperplan à séparer deux classes de données, positives et négatives.
- ✓ Les fonctions noyau (Kernel functions) sont utilisées pour la séparation non linéaire
- ✓ SVM a non seulement une base théorique rigoureuse, mais effectue également la classification avec plus de précision que la plupart des autres méthodes, en particulier pour des données de grande dimension

Etapes de formulation de SVM

1

Formulation initiale de SVM

- Primal Form

2

Convertir SVM à une forme qu'on peut résoudre

- Dual form(formulation duelle)

3

Tolérer des erreurs

- Soft margin

4

Permettre l'hyperplan non linéaire

- Kernel functions

Etapes de formulation de SVM

1

Formulation initiale de SVM

- Primal Form

2

Convertir SVM à une forme qu'on peut résoudre

- Dual form(formulation duelle)

3

Tolérer des erreurs

- Soft margin

4

Permettre l'hyperplan non linéaire

- Kernel functions

SVM : **Classification Binaire: Concepts de base**

✓ Soit l'ensemble de données d'entrainement D suivant:

$\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots\ldots, (\boldsymbol{x_n}, y_n)\}$

Où $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \ldots\ldots\ldots, x_{ir})$ est un **vecteur input**

et y_i est sa **classe label** (target) , $y_i \epsilon \{-1,1\}$

1: classe positive et -1: classe négative

✓ SVM trouve une fonction linéaire qui sépare les données en deux classes

$$f(\boldsymbol{x}) = \boldsymbol{w^T x} + b \qquad avec\ w: vecteur\ des\ paramètres$$

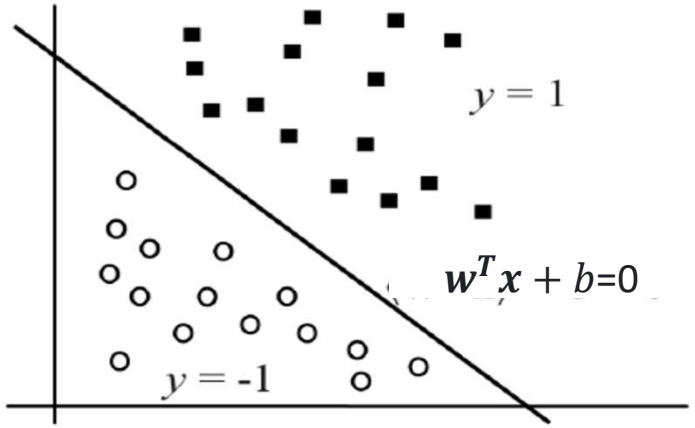
$$y_i = \begin{cases} 1 \ \ si \quad \boldsymbol{w^T x_i} + b \geq 0 \\ -1 \ \ si \quad \boldsymbol{w^T x_i} + b < 0 \end{cases}$$

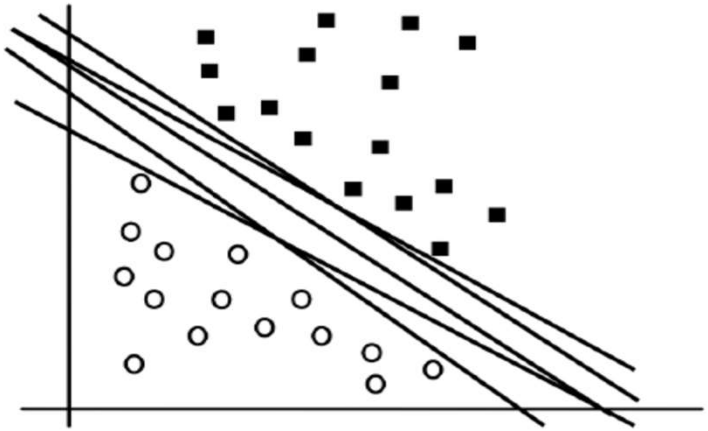
SVM : **Hyperplan: classifieur binaire**

- ✓ Quand les données sont linéairement séparables
- ✓ L'hyperplan qui sépare les données d'entrainement en deux classes (dites positive et négatives) est

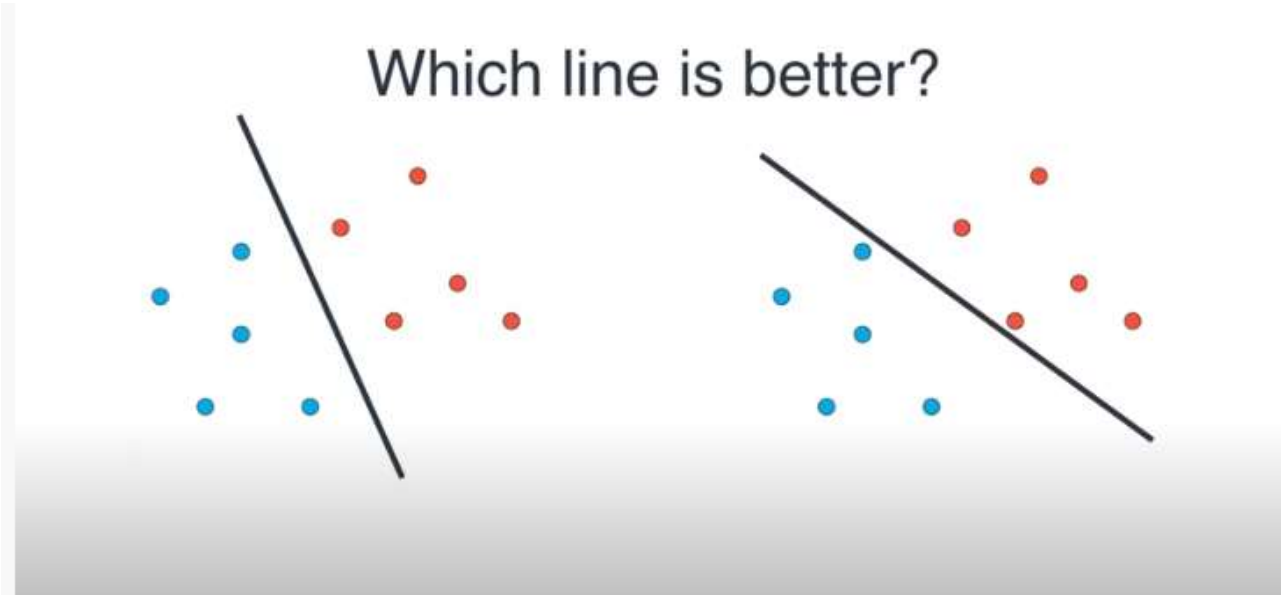
$$\boldsymbol{H}: \quad \boldsymbol{w}^T\boldsymbol{x} + b=0$$

- ✓ Il est aussi appelé la frontière de décision
- ✓ Plusieurs hyperplans sont possibles, quel est le meilleur?

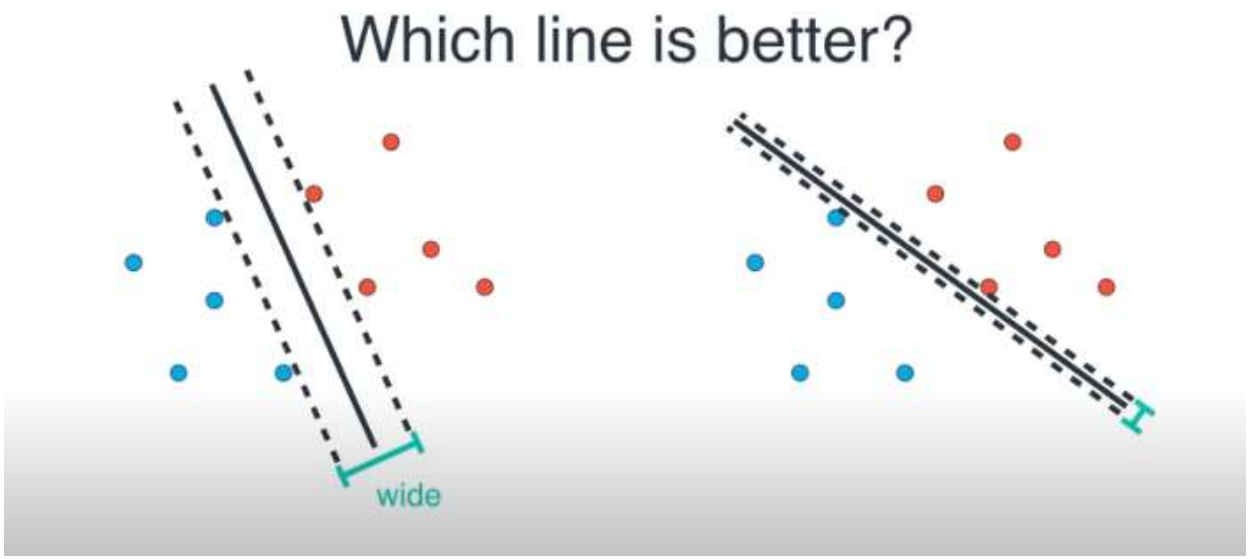




SVM : **Hyperplan- Illustration**

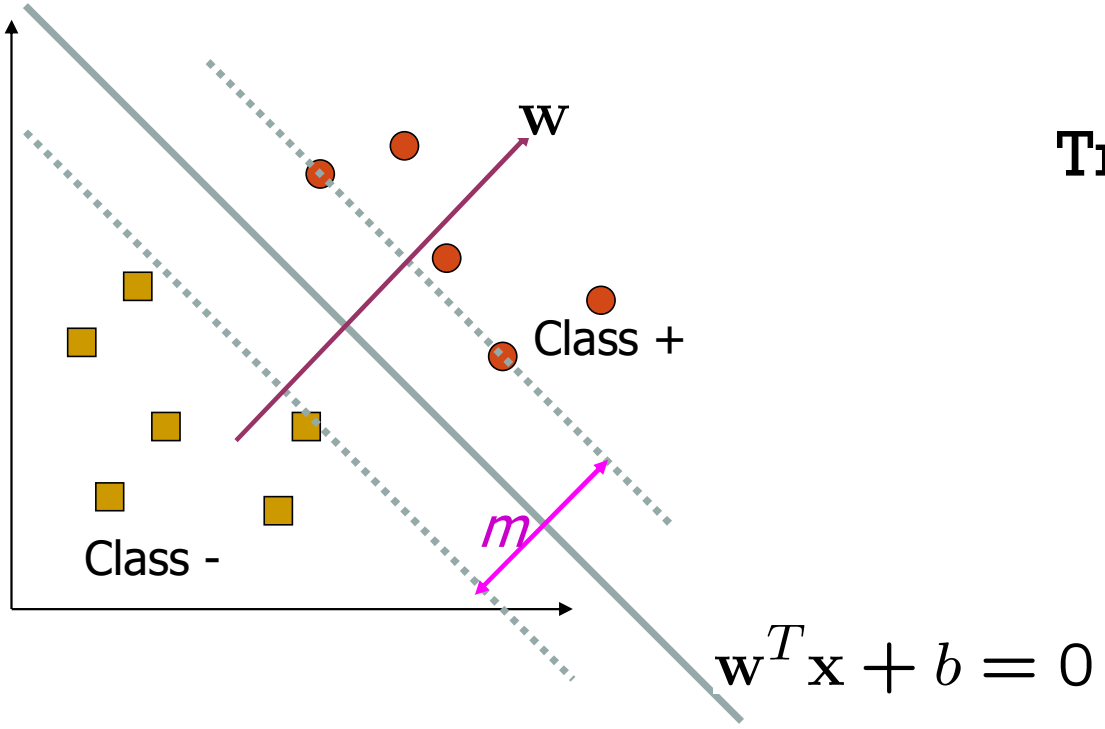


SVM : **Hyperplan- Illustration**



SVM : Recherche de la solution optimale: Quel Hyperplan choisir?

- ✓ Le meilleur hyperplan est celui qui **maximise la marge** entre les deux classes
- ✓ Cet hyperplan **minimise l'erreur** de classification **et réduit l'overfitting**



Trouver **m**

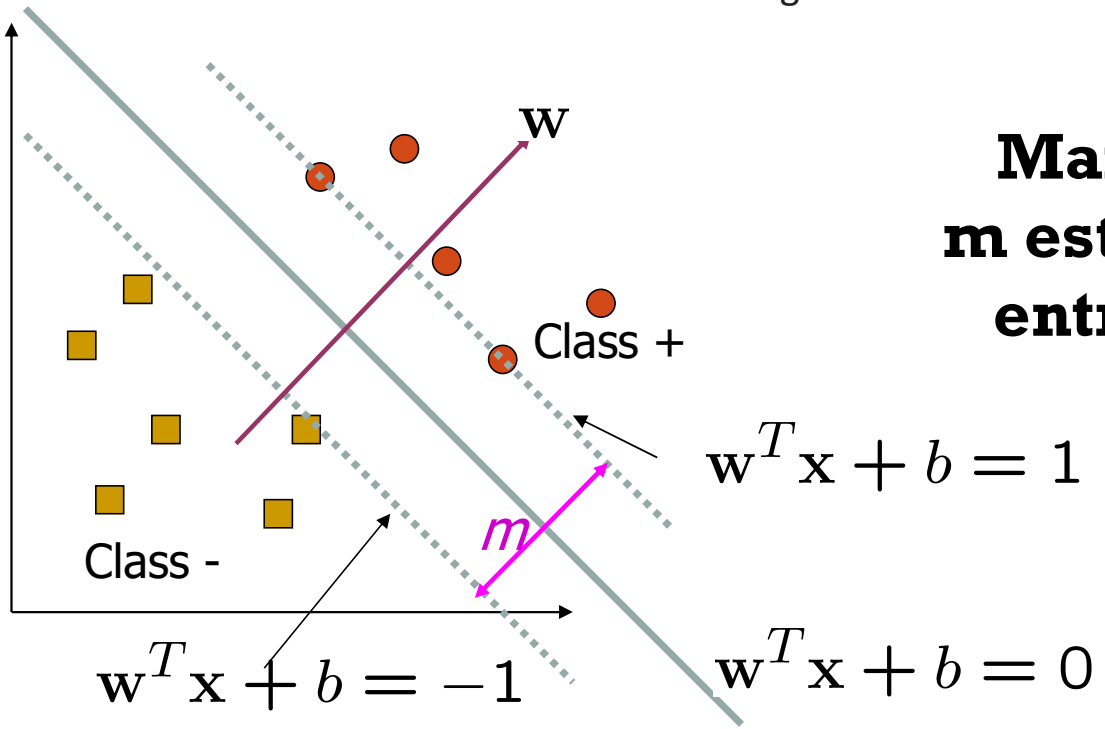
m=?

SVM : **Hyperplan avec la marge maximale**

Pour un hyperplan H quelconque d'équation H: $w^T x + b$=0, il existe deux hyperplans parallèles et équidistants de H qui sont

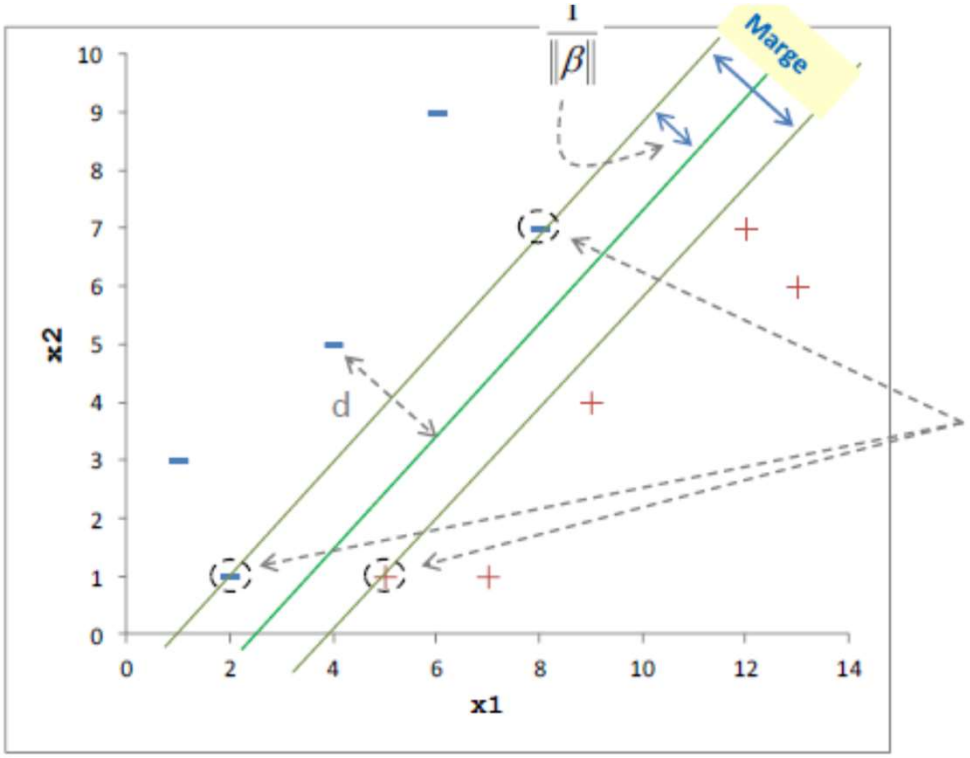
$H+$: $w^T x + b$=1 H+ sur le bord de la classe positive

$H-$: $w^T x + b$=-1 H- sur le bord de la classe négative



Maximiser m
m est la distance entre H+ et H-

SVM : **Les points Support Vecteurs**

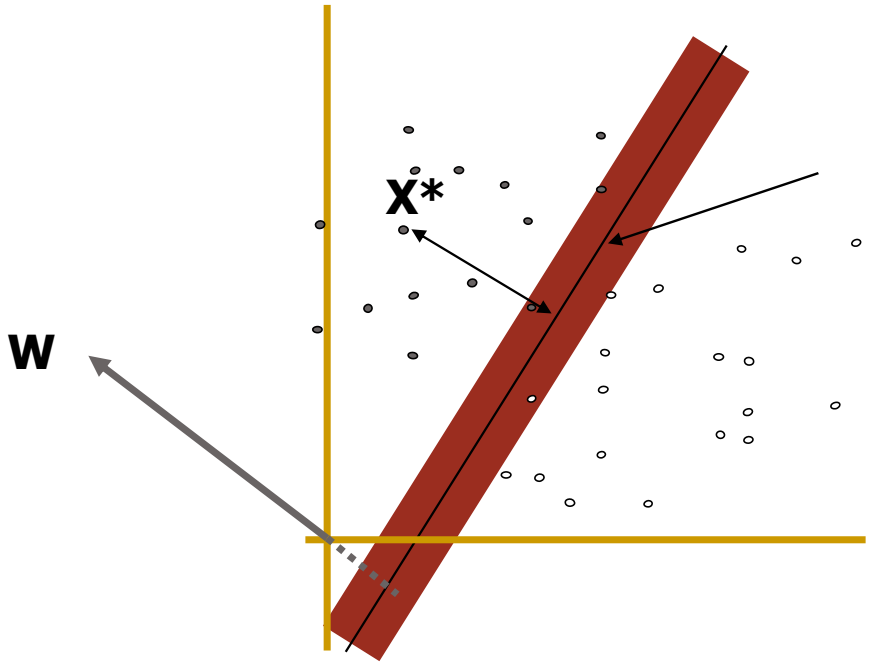


Les points où «s'appuient» les droites «marges» sont les «vecteurs de support». Si on les retire de l'échantillon, la solution est modifiée.

- Plusieurs zones sont définies dans l'espace de représentation

f(x) = 0, on est sur la frontière
f(x) > 0, on classe «+»
f(x) < 0, on classe «-»
f(x) = +1 ou -1, on est sur les droites délimitant des vecteurs de support

SVM : **Calculer la marge : Règle**



$\mathbf{w^T x} + b = 0$

x – Vector

W – Normal Vector

b – Scale Value

- Quelle est la distance orthogonale entre un point x^* quelconque **à** une droite $\mathbf{w^T x}+b= 0$?

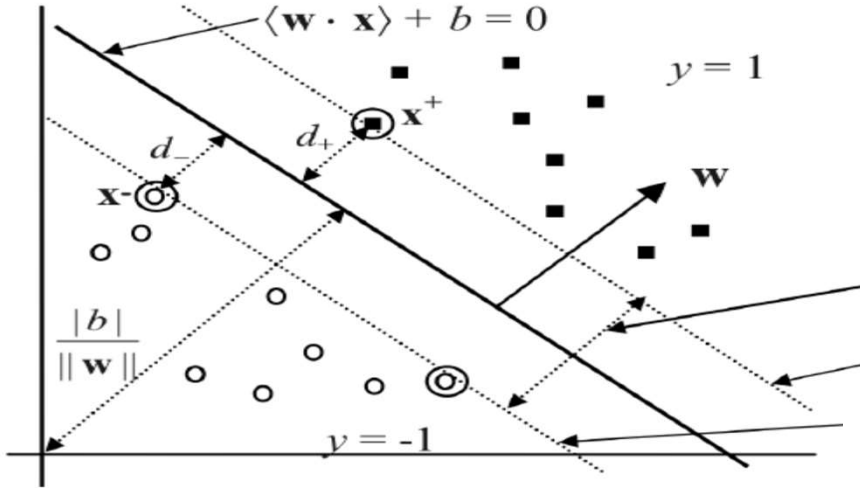
$$d(x) = \frac{|w^T x^* + b|}{\|w\|} = \frac{|w^T x^* + b|}{\sqrt{\sum_{i=1}^{r} w_i^2}}$$

SVM : **calculer la marge m**

$$m = d(x^+, H) + d(x^-, H)$$

$$= \frac{|w^T x^+ + b|}{\|w\|} + \frac{|w^T x^- + b|}{\|w\|} = \frac{1}{\|w\|} + \frac{1}{\|w\|} \quad \rightarrow \quad \boldsymbol{m = \frac{2}{\|w\|}}$$

Maximiser m revient à **minimiser** $\|w\|$



$H+$: $w^T x + b$=1

$H-$: $w^T x + b$=-1

SVM : Un problème d'optimisation: Formulation initiale

Il s'agit d'un **problème d'optimisation quadratique avec contrainte**

$$minimiser \quad \frac{1}{2}\|w\|^2$$

Sous contrainte $y_i(w^T x_i + b) \geq 1 \quad \forall i$

$\boldsymbol{y_i(w^T x_i + b) \geq 1 \quad \forall i}$ *signifie que*

$(w^T x_i + b) \geq 1$ **pour** y_i=1

$(w^T x_i + b) \leq -1$ **pour** y_i=-1

SVM : **Formulation initiale: Inconvénients**

$$minimiser \quad \frac{1}{2}\|w\|^2$$
$$\textbf{Sous contrainte } y_i(w^T x_i + b) \geq 1 \quad \forall i$$

(1)Les algorithmes d'optimisation numérique (prog. quadratique) ne sont pas opérationnels dès que le nombre de paramètres «p» est grand (> quelques centaines). Ce qui arrive souvent dans le traitement des données complexes (textmining, image, …) (peu d'exemples, beaucoup de descripteurs)

(2)Cette écriture ne met pas en évidence la possibilité d'utiliser des fonctions «noyau» qui permettent d'aller au-delà du cadre des classifieurs linéaires

Etapes de formulation de SVM

1

Formulation initiale de SVM

- Primal Form

2

Convertir SVM à une forme qu'on peut résoudre

- Dual form(formulation duale)

3

Tolérer des erreurs

- Soft margin

4

Permettre l'hyperplan non linéaire

- Kernel functions

Expression duale passage au Lagangien

Un problème d'optimisation possède Une forme duale si on évolue dans un Espace convexe. ce qui est le cas, On passe alors par le Lagrangien

Le problème initial…

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

Sous contrainte $y_i(w^T x_i + b) \geq 1 \quad \forall i = 1, \dots, n$ (1)

devient sous sa forme duale

$$min \ L_d(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1) \quad (2)$$

La théorie de l'optimisation dit qu'une solution optimale à (1) doit satisfaire cinq conditions, appelées conditions de Karush-Kuhn-Tucker,

- $\alpha_i \geq 0$ sont les multiplicateurs de Lagrange
- avec $\alpha_i(1 - y_i (w^T x_i + b)) = 0$
- et $(y_i (w^T x_i + b) - 1) \geq 0$
- $\frac{dL}{dw} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0$
- $\frac{dL}{db} = \sum_{i=1}^{n} \alpha_i y_i = 0$

La forme duale

En introduisant les informations issues de l'annulation des dérivées partielles du Lagrangien, on obtient une optimisation ne dépendant que des multiplicateurs α_i

- La nouvelle fonction objective est exprimée uniquement par α_i (Lagrange multipliers)
- Remplacer w par sa formule issue de la dérivée
- Pour plus de details sur le fondement mathématique, visiter ce lien

http://www.adeveloperdiary.com/data-science/machine-learning/support-vector-machines-for-beginners-duality-problem/

$$\max. \; \mathbf{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \qquad (3)$$

sachant que $\alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$

Charactéristiques de la Solution du problème dual

- Plusieurs α_i sont zero
  - **w** est une combinaison linéaire d'un petit nombre de α_i

    Comme nous avons la condition suivante de l'équation2:

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

- $\mathbf{x}_i$ avec α_i non nuls sont appelés support vectors (SV)
  - L'hyperplan de décision est determiné uniquement par SV
  - Soient *SV* les indices du *s* support vectors.
  - **w** *est exprimé ainsi*

$$\boldsymbol{w} = \sum_{\boldsymbol{i}\in \boldsymbol{SV}} \boldsymbol{\alpha_i y_i\, x_i}$$

Des points supports aux coefficients de l'hyperplan (I)

Puisque les expressions primales et duales sont deux facettes du même problème, on doit pouvoir passer de l'un à l'autre.

A partir de la dérivée partielle du Lagrangien par rapport à *w*

$$\frac{dL}{dw} = w - \sum_{i=1}^{n} \alpha_i \, y_i \, x_i = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{n} \alpha_i \, y_i \, x_i$$

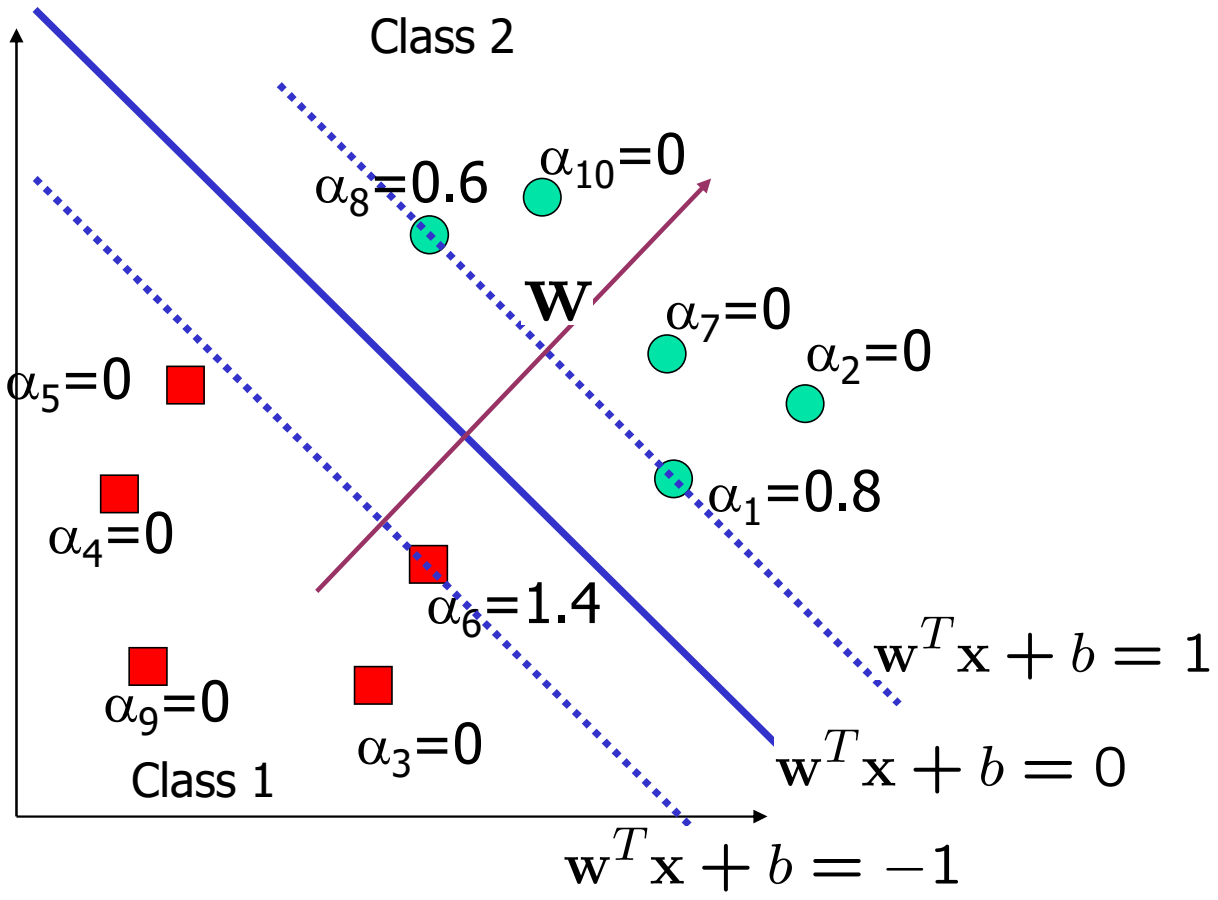
Seuls les points supports participent au calcul des coefficients, puisque ce sont les seuls pour lesquels ($\alpha_i > 0$)

On peut obtenir b (pour les ($\alpha_i > 0$))

$$\alpha_i (1 - y_i \, (w^T x_i + b)) = 0$$

$$b = \frac{1 - y_i w^T x_i}{y_i}$$

Interpretation Géométrique



Trouver les coefficients de l'hyperplan optimal

Exemple numérique

Dans cet exemple β $c'est\ le\ vecteur\ de\ paramètres\ w$

$w_1 = \beta_1$ $et\ b = \beta_0$

Pour β_1, seule la variable X_1 prend part aux calculs

$$\begin{aligned}\beta_1 &= 0.333\times(-1)\times 2 \\ &+ 0.111\times(-1)\times 8 \\ &+ 0.444\times(1)\times 5 \\ &= 0.6667\end{aligned}$$

Vecteurs de support

n°	x1	x2	y	alpha
2	2	1	-1	**0.333**
5	8	7	-1	**0.111**
6	5	1	1	**0.444**

beta.1	0.6667
beta.2	-0.6667

beta.0	-1.6667
	-1.6667
	-1.6667

$$\begin{aligned}\beta_0 &= \frac{1-y_i x_i^T \beta}{y_i} \\ &= \frac{1-(-1)\times[0.667\times 2+(-0.667)\times 1]}{(-1)} \\ &= -1.6667\end{aligned}$$

Le résultat est le même quel que soit le point support utilisé.

Si on choisit le point support n°2

Formule finale d'hyperplan H

- ***Equation d'hperplan de decision:***

$$H\colon w^T x + b = \sum_{i \in SV} y_i \alpha_i {x_i}^T x + b$$

- ***Testing:***

Soit z une instance de test

$$\boldsymbol{sign}(\, w^T z + b) = \boldsymbol{sign}(\textstyle\sum_{i \in SV} y_i \alpha_i {x_i}^T z + b)$$

Si le résultat est positif alors z est classifié dans la classe positive.

Sinon, z est classée négative

Classement d'un individu supplémentaire (II)

Exemple numérique

Pour le classement de l'individu n°1

$$f(x) = \sum_{i \in S} \alpha_i y_i \langle x_i, x \rangle + \beta_0$$
$$= 0.333 \times (-1) \times (2 \times 1 + 1 \times 3) + 0.111 \times (-1) \times (8 \times 1 + 7 \times 3) + 0.444 \times (1) \times (5 \times 1 + 1 \times 3) + (-1.667)$$
$$= -3.0$$

n°	x1	x2	y	f(x)	prediction
1	1	3	-1	-3.000	-1
2	2	1	-1	-1.000	-1
3	4	5	-1	-2.333	-1
4	6	9	-1	-3.667	-1
5	8	7	-1	-1.000	-1
6	5	1	1	1.000	1
7	7	1	1	2.333	1
8	9	4	1	1.667	1
9	12	7	1	1.667	1
10	13	6	1	3.000	1

Utilisation des 3 points supports.

Vecteurs de support

n°	x1	x2	y	alpha
2	2	1	-1	**0.333**
5	8	7	-1	**0.111**
6	5	1	1	**0.444**

Beta.0	-1.667

Formulation Duale

- Cette écriture est complètement cohérente avec la formulation primale
- Elle met mieux en lumière le rôle des points supports avec les pondérations α_i
- Elle met en lumière également le rôle fondamental du produit scalaire <xi,xi'> dans les calculs, ce sera très important par la suite (cf. fonctions «noyau»)
- Dans les très grandes dimensionnalités (''p'' très élevé, ''n'' relativement faible –traitement d'images, textmining), cette écriture rend les calculs plus simples pour les techniques d'optimisation

Etape Suivante

1

Convertir SVM à une forme qu'on peut résoudre

- Dual form(forme duelle)

2

Tolérer des erreurs

- Soft margin

3

Permettre l'hyperplan non linéaire

- Kernel functions

4

SVM multi classes

Rappel

- Quand les données sont linéairement séparables, le problème était de

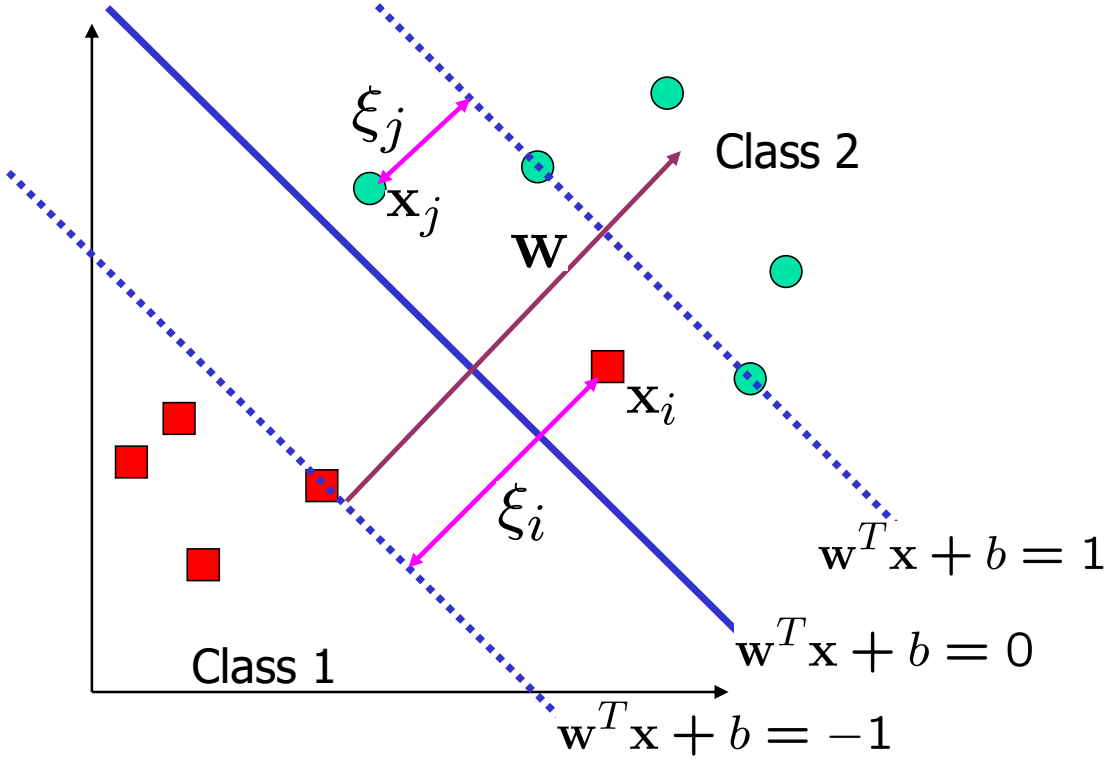
$$minimiser \quad \frac{1}{2}\|w\|^2$$

Avec la contrainte $y_i(w^T x_i + b) \geq 1 \quad \forall i$

- Pour réduire le problème d'overfitting , il est judicieux de permettre des erreurs

Tolérer les erreurs dans la solution

- Nous tolérons "l'erreur" ξ_i en classification; la tolerance à l'erreur doit figurer dans l'output de la fonction $\mathbf{w}^T\mathbf{x}+b$
- ξ_i approxime le nombre des exemples mal classifies,



Soft Marge d'Hyperplan

- Si nous minimisons $\sum_i \xi_i$, ξ_i peut être trouvé par

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

  - ξ_i sont "slack variables" en optimisation
  - Note que ξ_i=0 si pas d'erreur pour $\mathbf{x}_i$
- Nous minimisons

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

  - C : constante cost parameter
- Le problème d'optimization devient

$$\text{Minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{Avec la contrainte } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Formulation de l'optimisation

Formulation Primale

$$\text{Minimize } \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

Sous la contrainte $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$

Formulation duale

$$\max_{\alpha} L_D(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{i'=1}^{n}\alpha_i\alpha_{i'}y_iy_{i'}\langle x_i, x_{i'}\rangle$$

$$s.c.$$

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

$$0 \leq \alpha_i \leq C, \forall i$$

La tolérance aux erreurs est plus ou moins accentuée avec le paramètre **C** (''cost'' parameter)

➔ C trop élevé, danger de sur-apprentissage

➔ C trop faible, sous-apprentissage

Etape Suivante

1

Convertir SVM à une forme qu'on peut résoudre

- Dual form(forme duelle)

2

Tolérer des erreurs

- Soft margin

3

Permettre l'hyperplan non linéaire

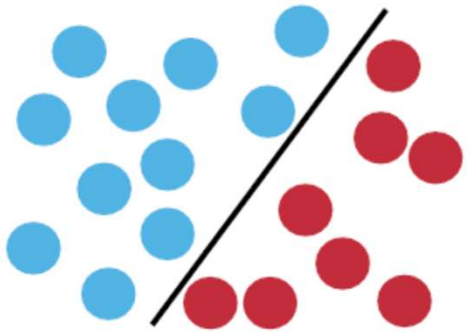
- Kernel functions

4

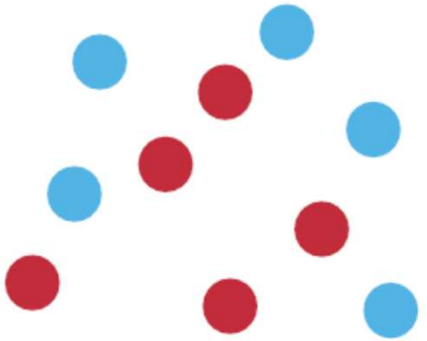
La SVM multi classes

Extension au frontière Non linéaire

- Et si les données ne sont pas linéairement séparables?
- Comment généraliser le SVM pour le cas non linéaires?



Linear SVM can function well for linear separable datasets.



Nonlinear datasets cannot be split with straight lines.

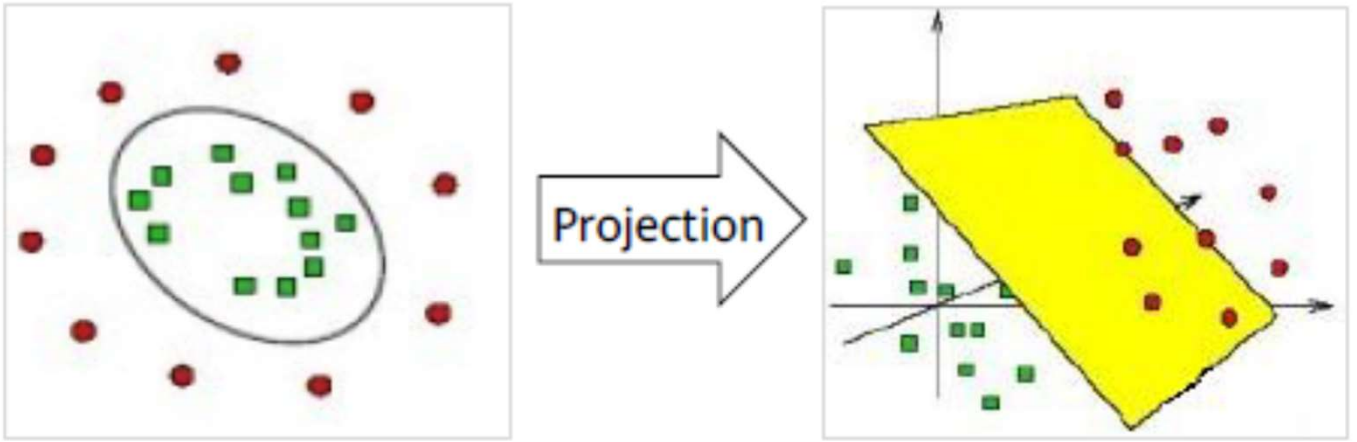
Extension au frontière Non linéaire

- Pour couvrir la separation non linéaire, les mêmes formules et techniques sont utilisées,
- Il faut juste transformer les données en un autre espace (généralement de plus grande dimension),

  ⟹ Ce qui permet une séparation linéaire dans le nouvel espace de données transformé,

  - Input space: l'espace des données originales $\mathbf{x}_i$
  - Feature space: L'espace des données transformées $\phi(\mathbf{x}_i)$

Extension au frontière Non linéaire- Illustration



Complex segmentation in low-dimensional space

Easy segmentation in high-dimensional space

Espace transformé

- L'idée de base est de transformer les données d'Entrée de l'espace X vers l'espace de features F via une function non linéaire ϕ,

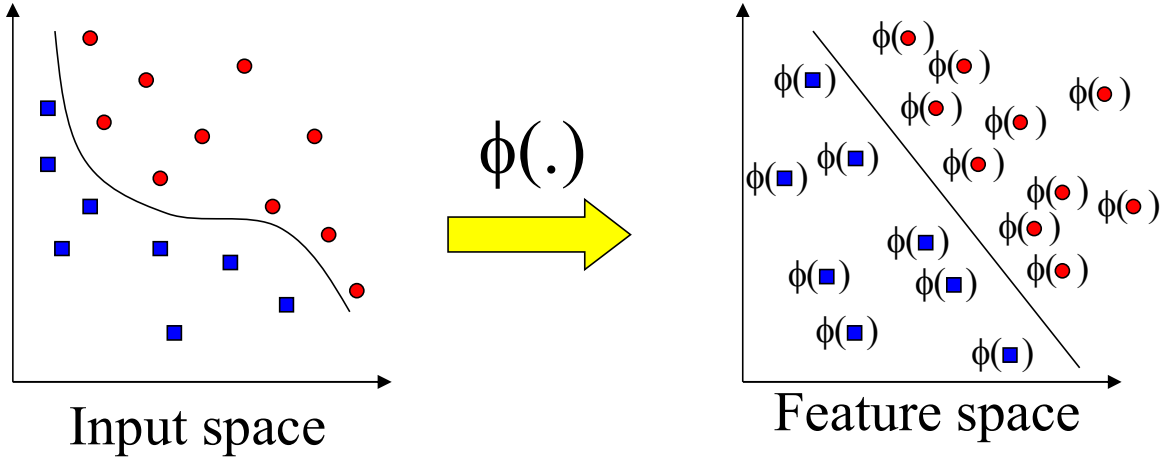
$$\phi : X \rightarrow F$$
$$\mathbf{x} \mapsto \phi(\mathbf{x})$$

- Après la transformation, les données d'entrée originales

$\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots., (\boldsymbol{x_n}, y_n)\}$ devient:

$\{(\boldsymbol{\phi}(\boldsymbol{x_1}), y_1), (\boldsymbol{\phi}(\boldsymbol{x_2}), y_2), \ldots., (\boldsymbol{\phi}(\boldsymbol{x_n}), y_n)\}$

Transformation des données



Note: feature space is of higher dimension than the input space in practice

- Dans cet exemple, l'espace transformé est aussi 2-D,
- Mais, en general, le nombre de dimensions dans l'espace des features est plus grand que dans l'espace d'entrée,

Transformation des données

Le problème d'optimisation devient:

minimiser $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$

Sous la contrainte $y_i(w^t\phi(x_i) + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0$

Sa forme duale correspond à:

$$\max_{\alpha} L_D(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{i'=1}^{n}\alpha_i\alpha_{i'}y_iy_{i'}K(x_i, x_{i'})$$

$$s.c.$$

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

$$0 \leq \alpha_i \leq C, \forall i$$

La fonction de décision devient:

$$H\colon w^T\phi(x) + b = \sum_{i\in SV} y_i\alpha_i\phi(x_i)^T\phi(x) + b$$

Astuce noyau (the Kernel Trick)

- La transformation d'espace est très couteuse, puisqu'on doit calculer la function Φ pour chaque donnée x_i,
- Cependant, on peut s'en passer du clacul explicite de Φ en l'exprimant sous forme d'un produit vectoriel
- On définit alors la fonction kernel *K* par

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

Un exemple pour ϕ(.) et K(.,.)

- Suppose ϕ(.) est donné comme suit

$$\phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- Un produit vectoriel dans l'espace feature est

$$\langle \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}), \phi(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}) \rangle = (1 + x_1y_1 + x_2y_2)^2$$

- Alors, si on définit la function kernel comme suit on n'a plus besoin de calculer ϕ(.) explicitement

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- L'utilisation de kernel function (function noyau) pour éviter le calcul de ϕ(.) explicitement est connu par "kernel trick" (l'astuce de noyau)

Exemples de Kernel Functions

- kernel polynomial de degré *d*

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- Radial basis function (RBF) kernel avec largeur σ

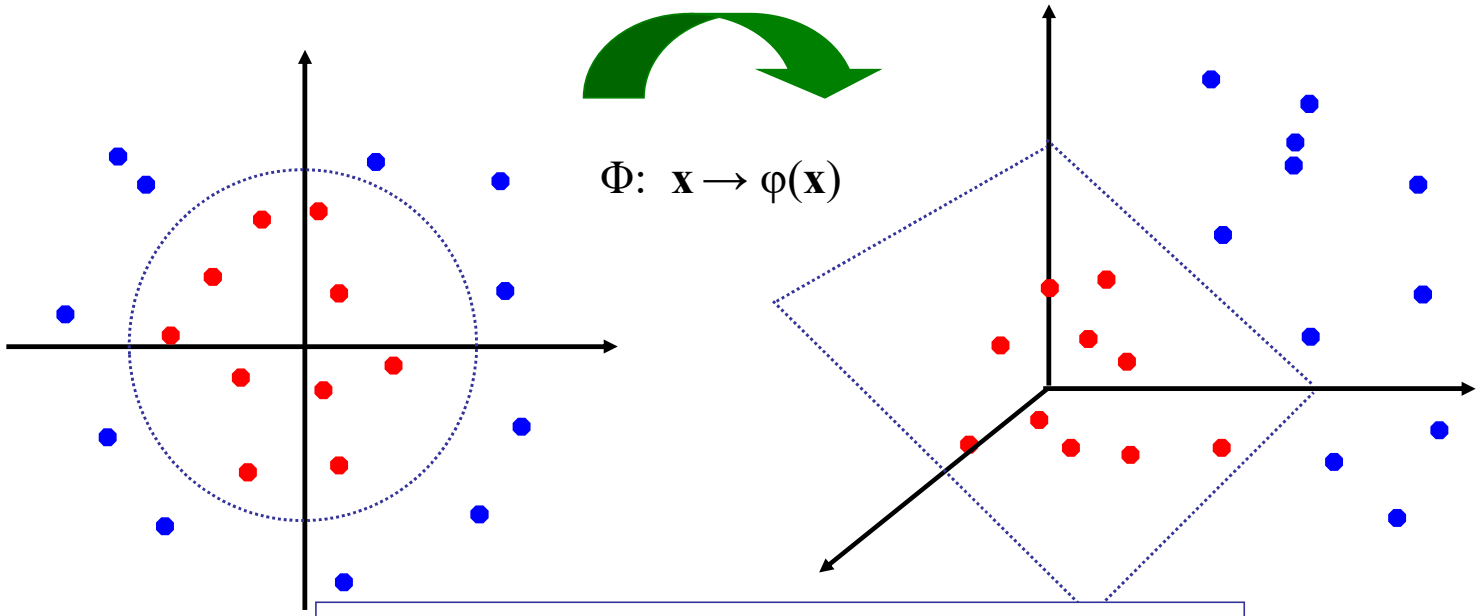
$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

- Sigmoid kernel with parameter κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

Exemple

- Suppose we have 5 one-dimensional data points
  - x_1=1, x_2=2, x_3=4, x_4=5, x_5=6, with 1, 2, 6 as class 1 and 4, 5 as class 2 $\Rightarrow$ y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1
- We use the polynomial kernel of degree 2
  - $K(x,y) = (xy+1)^2$
  - C is set to 100
- We first find α_i (i=1, …, 5) by

$$\text{max.} \sum_{i=1}^{5} \alpha_i - \frac{1}{2} \sum_{i=1}^{5} \sum_{i=1}^{5} \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^{5} \alpha_i y_i = 0$$

Exemple

- By using a QP (Quadratic Programming) solver, we get
  - α_1=0, α_2=2.5, α_3=0, α_4=7.333, α_5=4.833
  - Note that the constraints are indeed satisfied
  - The support vectors are {x_2=2, x_4=5, x_5=6}
- The discriminant function is

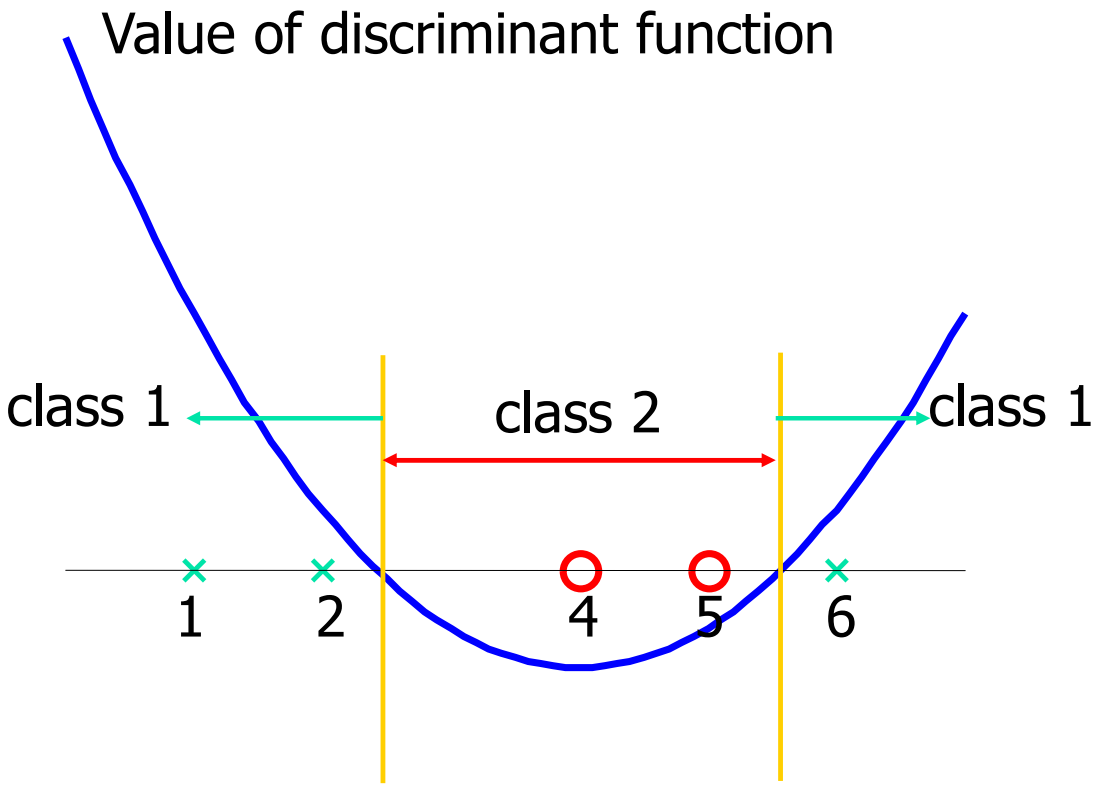
$$\begin{aligned} & f(z) \\ &= 2.5(1)(2z+1)^2 + 7.333(-1)(5z+1)^2 + 4.833(1)(6z+1)^2 + b \\ &= 0.6667z^2 - 5.333z + b \end{aligned}$$

(α_5, y_5, $K(z, x_5)$)

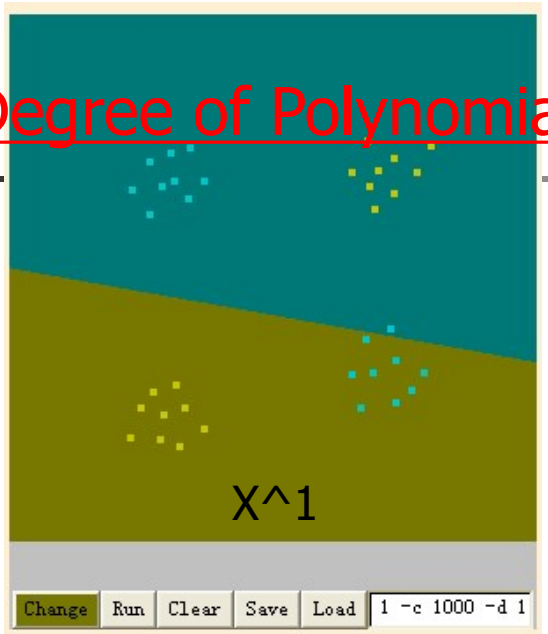
- *b* is recovered by solving f(2)=1 or by f(5)=-1 or by f(6)=1, as x_2 and x_5 lie on the line $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = 1$ and x_4 lies on the line $\phi(\mathbf{w})^T\phi(\mathbf{x}) + b = -1$
- All three give b=9

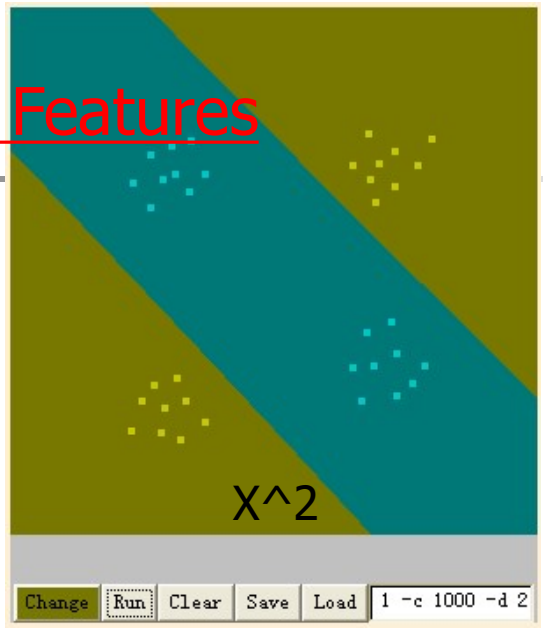
$$\rightarrow f(z) = 0.6667z^2 - 5.333z + 9$$

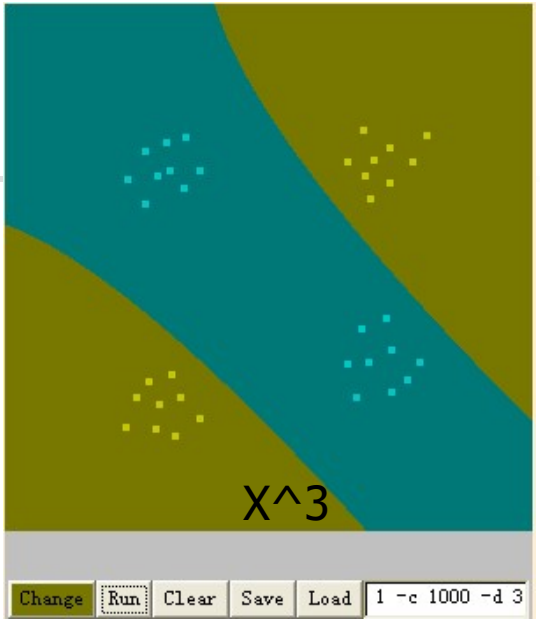
Example

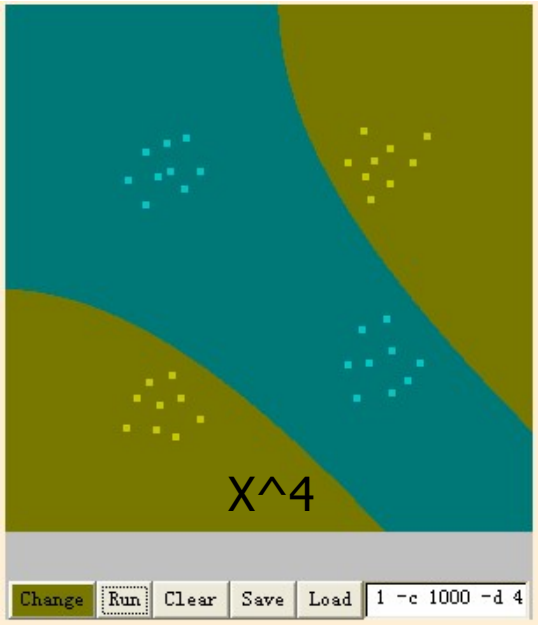

Value of discriminant function
class 1
class 2
class 1
1
2
4
5
6

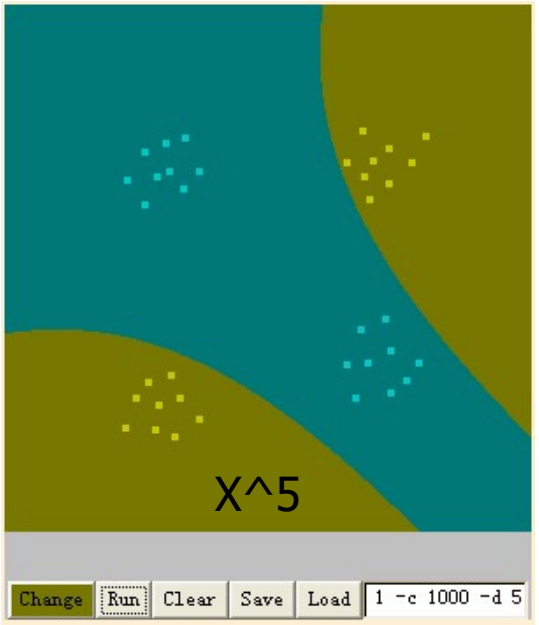
Degree of Polynomial Features

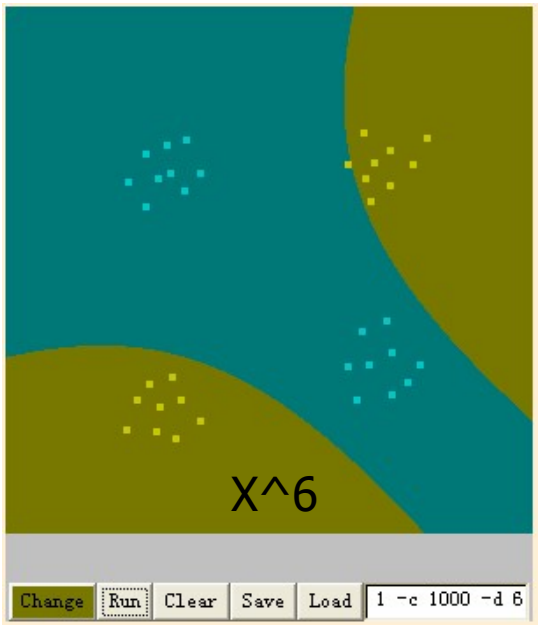












Etape Suivante

1

Convertir SVM à une forme qu'on peut résoudre

- Dual form(forme duelle)

2

Tolérer des erreurs

- Soft margin

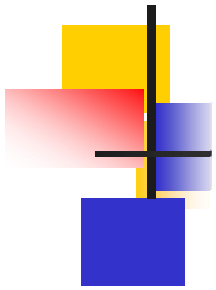
3

Permettre l'hyperplan non linéaire

- Kernel functions

4

SVM multi classes



SVM Multiclasses

Approche ''one-against-rest''

L'approche SVM est formellement définie pour les problèmes à deux classes, comment l'étendre (simplement) à des problèmes à K classes avec $Y \in \{y_1,\ldots,y_K\}$?

- Construire tour à tour K modèles discriminants avec la modalité y_k contre les autres ($Y' \in \{y_k=+1, y_{(k)}=-1\}$). Nous obtenons K fonctions de décision $f_k(x)$
- En classement, prédire la classe présentant le score le plus élevé c.-à-d.

$$\hat{y} = \arg\max_k f_k(x)$$

On retrouve le schéma bayésien avec la maximisation de la probabilité a posteriori

- **Caractéristiques** : K apprentissages à effectuer sur les données
- **Avantages** : simplicité
- **Inconvénients** : on peut introduire artificiellement un déséquilibre des classes dans la construction des modèles individuels, si les scores sont mal calibrés, les comparaisons des output des fonctions de décision ne sont pas équitables

SVM Multiclasses

Approche « one vs. one » (pairwise)

- Construire tour à tour K(K-1)/2 modèles discriminant chaque paire de classes ($Y' \in \{y_k=+1, y_j=-1\}$). Nous obtenons K(K-1)/2 fonctions $f_{k,j}(x)$
- En classement, prédire la classe en utilisant un système de vote c.-à-d. présentant le plus grand nombre de victoires

$D_k(x)$ va fournir le "#votes" de la modalité y_k
Sachant que $f_{j,k}(x) = -f_{k,j}(x)$

La règle d'affectation s'appuie sur le #vote max.

$$D_k(x) = \sum_{j \neq k, j=1}^{K} \text{sign}\left[f_{k,j}(x)\right]$$

$$\hat{y} = \arg\max_k D_k(x)$$

- **Remarque** : en cas d'ex-aequo (2 classes ou plus ont le même nombre de votes), on procède à la somme des scores $f_{k,j}(x)$ et on prend le max
- **Caractéristiques** : K(K-1)/2 apprentissages à effectuer, mais avec des ensembles de données de taille réduite
- **Avantages** : moins de problème de déséquilibre des classes, les scores sont mieux calibrés
- **Inconvénients** : temps de calcul quand K augmente

Bilan

Avantages

•Capacité à traiter de grandes dimensionnalités (#variables élevé)

•Robuste même quand le rapport ‘’#observations / #variables’’ est inversé

•Traitement des problèmes non linéaires avec le choix des noyaux

•Non paramétrique

•Robuste par rapport aux points aberrants (contrôlé avec le paramètre C)

•#points supports donne une bonne indication de la complexité du problème traité

•Souvent performantdans les comparaisons avec les autres approches

•Paramétrage permet de la souplesse (ex. résistance au sur-apprentissage avec C)

Inconvénients

•Difficulté à identifier les bonnes valeurs des paramètres(et sensibilité aux paramètres)

•Difficulté à traiter les grandes bases avec #observations très élevé (capacité mémoire avec matrice de Gram –si implémentation naïve, temps de calcul)

•Problème lorsque les classes sont bruitées (multiplication des points supports)

•Pas de modèle explicite pour les noyaux non linéaires (utilisation des points supports)

•Difficulté d’interprétations (ex. pertinence des variables)

•Le traitement des problèmes multi-classes reste une question ouverte