SVM : Machines à Vecteurs de Support ou Séparateurs à Vastes Marges

Mohamadally Hasan
Fomani Boris
BD Web, ISTY3
Versailles St Quentin, France
hmohamad@isty-info.uvsq.fr
bfomanik@isty-info.uvsq.fr

16 janvier 2006

Résumé : Nous présentons une description d'une méthode de classification par apprentissage particulière, les SVM. Etant donné que les algorithmes liés aux SVM sont composés de calculs et de fonctions mathématiques complexes, nous avons décidé de diviser la présentation en différentes parties, une destinée à un large public où nous décrivons de façon simple et assez complète les principes de fonctionnement et une destinée à un public plus ciblé où nous décrivons en détails l'aspect mathématiques des SVM. Nous intéressons aussi aux différents domaines d'application et nous insistons sur l'utilisation des machines à supports de vecteurs dans Oracle.

Mots-clés : Apprentissage supervisé, Induction, Classification, Séparateur à Vaste Marge, Support Vector Machine, Machine à Support de Vecteurs, Oracle Data Mining

Structure : Dans la section 1, nous présentons les SVM et nous effectuons un rappel sur la notion d'apprentissage. Ensuite dans la section 2, nous décrivons de manière générale le principe de fonctionnement des SVM. Les fondements mathématiques sont détaillés dans la section 3. Dans la section 4, nous nous intéressons aux différents domaines d'applications des SVM. Nous finissons par une présentation de l'implémentation de SVM dans Oracle dans la section 5

1 Introduction

Parmi les méthodes à noyaux, inspirées de la théorie statistique de l'apprentissage de Vladimir Vapnik, les SVM constituent la forme la plus connue. SVM est une méthode de classification binaire par apprentissage supervisé, elle fut introduite par Vapnik en 1995. Cette méthode est donc une alternative récente pour la classification. Cette méthode repose sur l'existence d'un classificateur linéaire dans un espace approprié. Puisque c'est un problème de classification à deux classes, cette méthode fait appel à un jeu de données d'apprentissage pour apprendre les paramètres du modèle. Elle est basée sur l'utilisation de fonction dites noyau (kernel) qui permettent une séparation optimale des données.

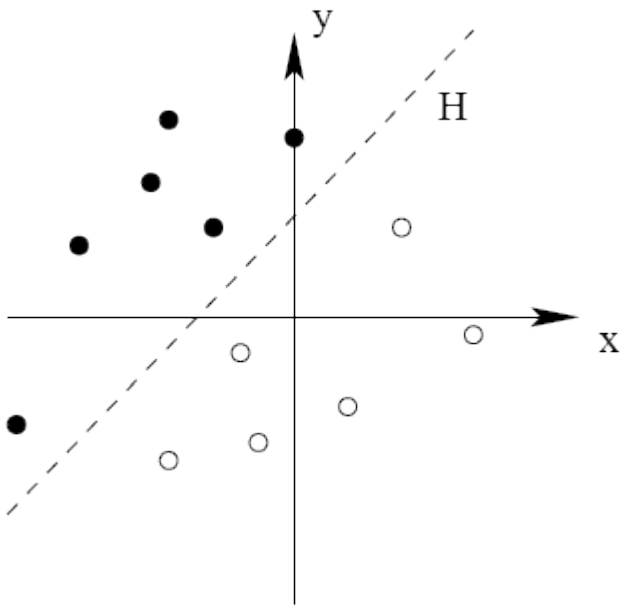
Dans la présentation des principes de fonctionnements, nous schématiserons les données par des « points » dans un plan.

La notion d'apprentissage étant importante, nous allons commencer par effectuer un rappel. L'apprentissage par induction permet d'arriver à des conclusions par l'examen d'exemples particuliers. Il se divise en apprentissage supervisé et non supervisé. Le cas qui concerne les SVM est l'apprentissage supervisé. Les exemples particuliers sont représentés par un ensemble de couples d'entrée/sortie. Le but est d'apprendre une fonction qui correspond aux exemples vus et qui prédit les sorties pour les entrées qui n'ont pas encore été vues. Les entrées peuvent être des descriptions d'objets et les sorties la classe des objets donnés en entrée.

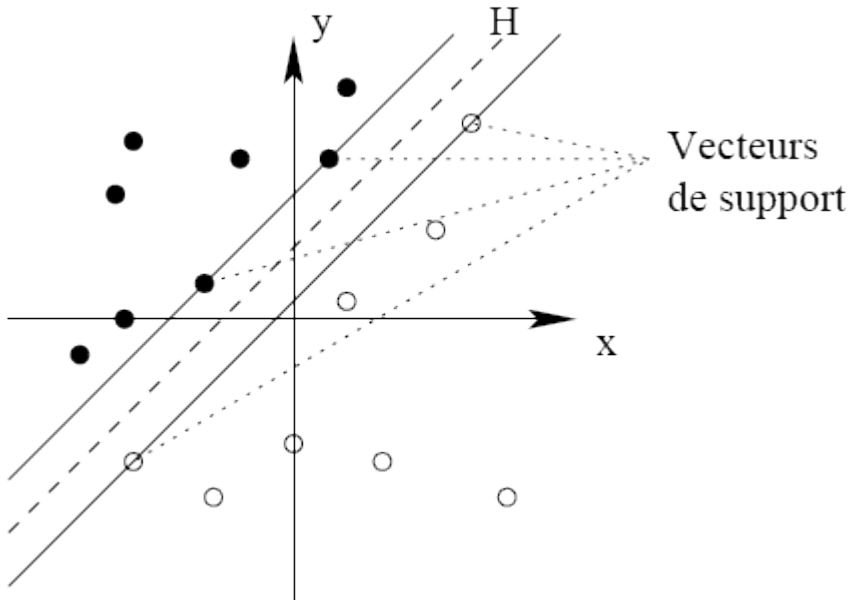
2 SVM principe de fonctionnement général

2.1 Notions de base : Hyperplan, marge et support vecteur

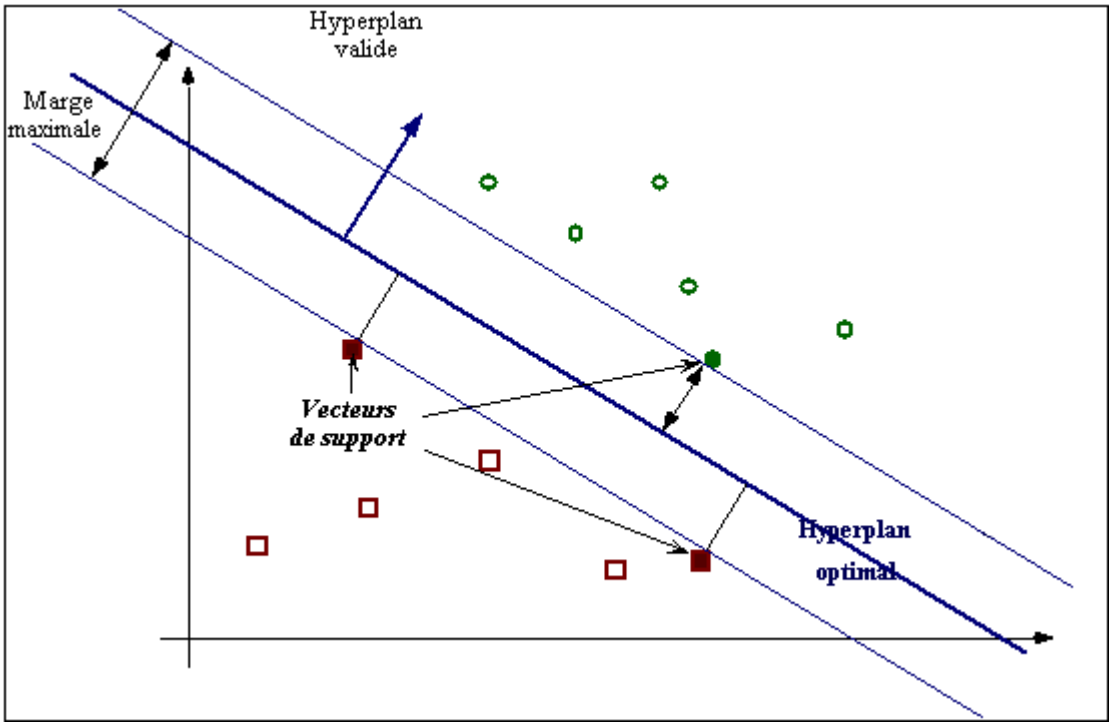
Pour deux classes d'exemples donnés, le but de SVM est de trouver un classificateur qui va séparer les données et maximiser la distance entre ces deux classes. Avec SVM, ce classificateur est un classificateur linéaire appelé hyperplan.
Dans le schéma qui suit, on détermine un hyperplan qui sépare les deux ensembles de points.



Les points les plus proches, qui seuls sont utilisés pour la détermination de l'hyperplan, sont appelés vecteurs de support.

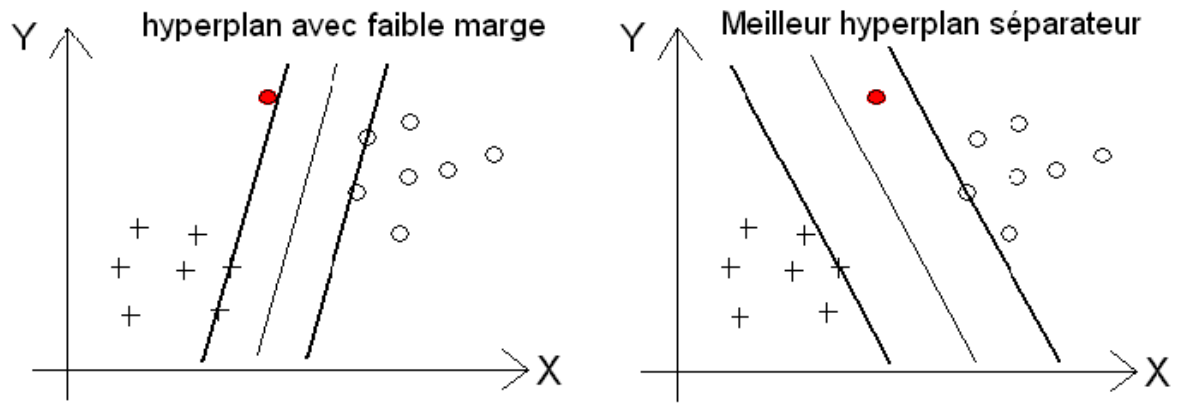


Il est évident qu'il existe une multitude d'hyperplan valide mais la propriété remarquable des SVM est que cet hyperplan doit être optimal. Nous allons donc en plus chercher parmi les hyperplans valides, celui qui passe « au milieu » des points des deux classes d'exemples. Intuitivement, cela revient à chercher l'hyperplan le « plus sûr ». En effet, supposons qu'un exemple n'ait pas été décrit parfaitement, une petite variation ne modifiera pas sa classification si sa distance à l'hyperplan est grande. Formellement, cela revient à chercher un hyperplan dont la distance minimale aux exemples d'apprentissage est maximale. On appelle cette distance « marge » entre l'hyperplan et les exemples. L'hyperplan séparateur optimal est celui qui maximise la marge. Comme on cherche à maximiser cette marge, on parlera de *séparateurs à vaste marge.*

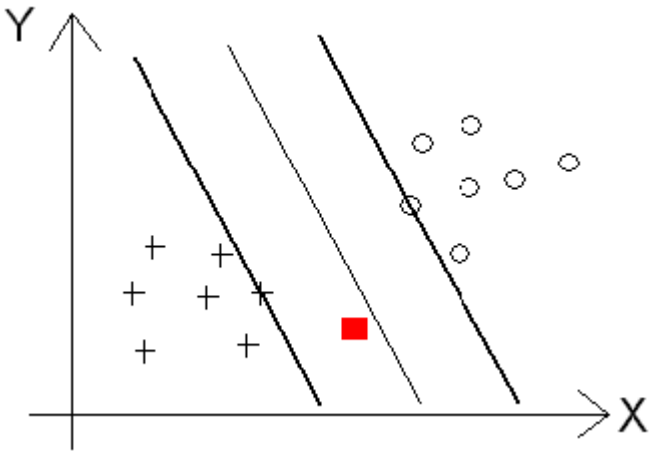


2.2 Pourquoi maximiser la marge ?

Intuitivement, le fait d'avoir une marge plus large procure plus de sécurité lorsque l'on classe un nouvel exemple. De plus, si l'on trouve le classificateur qui se comporte le mieux vis-à-vis des données d'apprentissage, il est clair qu'il sera aussi celui qui permettra au mieux de classer les nouveaux exemples. Dans le schéma qui suit, la partie droite nous montre qu'avec un hyperplan optimal, un nouvel exemple reste bien classé alors qu'il tombe dans la marge. On constate sur la partie gauche qu'avec une plus petite marge, l'exemple se voit mal classé.

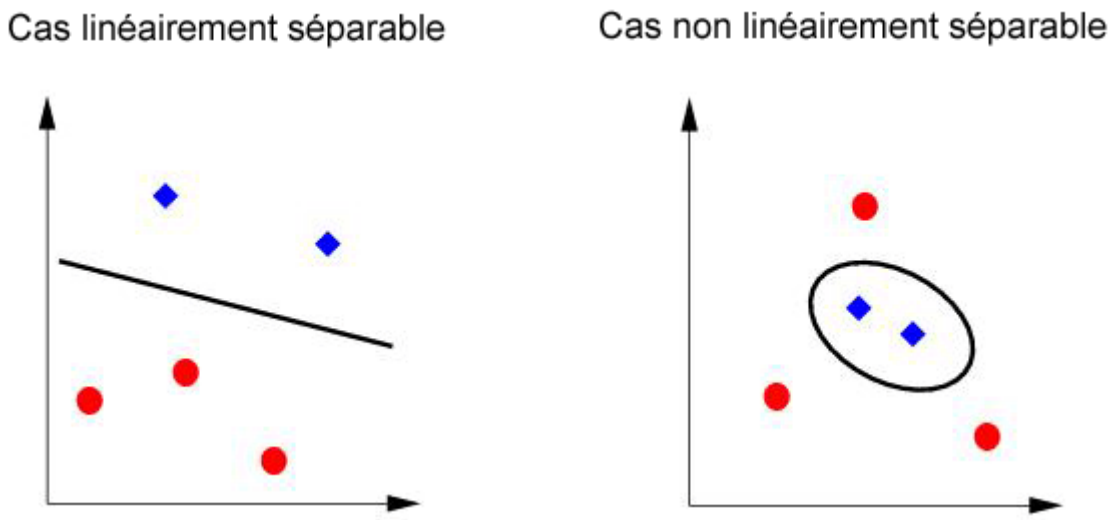


En général, la classification d'un nouvel exemple inconnu est donnée par sa position par rapport à l'hyperplan optimal. Dans le schéma suivant, le nouvel élément sera classé dans la catégorie des « + ».



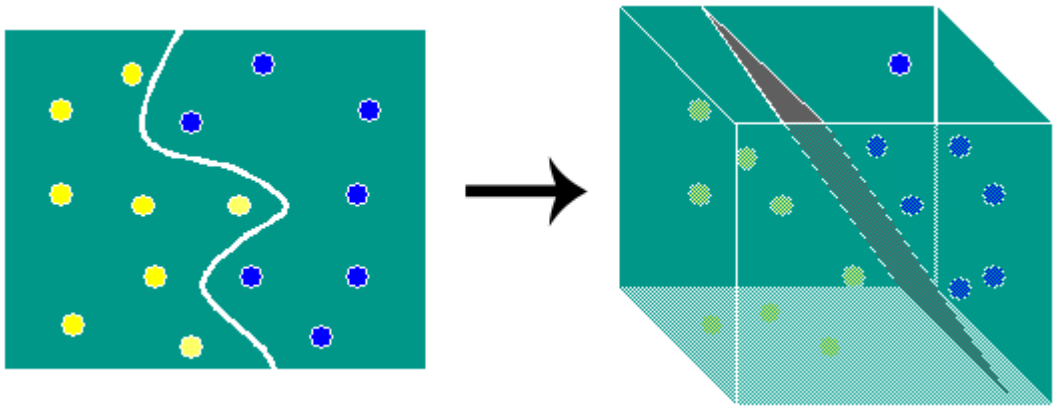
2.3 Linéarité et non-linéarité

Parmi les modèles des SVM, on constate les cas linéairement séparable et les cas non linéairement séparable. Les premiers sont les plus simple de SVM car ils permettent de trouver facilement le classificateur linéaire. Dans la plupart des problèmes réels il n'y a pas de séparation linéaire possible entre les données, le classificateur de marge maximale ne peut pas être utilisé car il fonctionne seulement si les classes de données d'apprentissage sont linéairement séparables.



2.4 Cas non linéaire

Pour surmonter les inconvénients des cas non linéairement séparable, l'idée des SVM est de changer l'espace des données. La transformation non linéaire des données peut permettre une séparation linéaire des exemples dans un nouvel espace. On va donc avoir un changement de dimension. Cette nouvelle dimension est appelé « espace de re-description ». En effet, intuitivement, plus la dimension de l'espace de re-description est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples est élevée. Ceci est illustré par le schéma suivant :

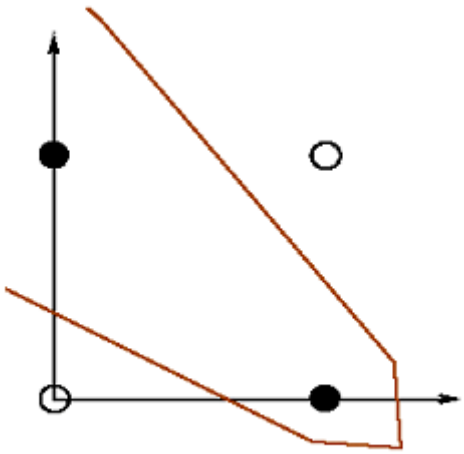


On a donc une transformation d'un problème de séparation non linéaire dans l'espace de représentation en un problème de séparation linéaire dans un espace de re-description de plus

grande dimension. Cette transformation non linéaire est réalisée *via* une fonction noyau. En pratique, quelques familles de fonctions noyau paramétrables sont connues et il revient à l'utilisateur de SVM d'effectuer des test pour déterminer celle qui convient le mieux pour son application. On peut citer les exemples de noyaux suivants : polynomiale, gaussien, sigmoïde et laplacien.

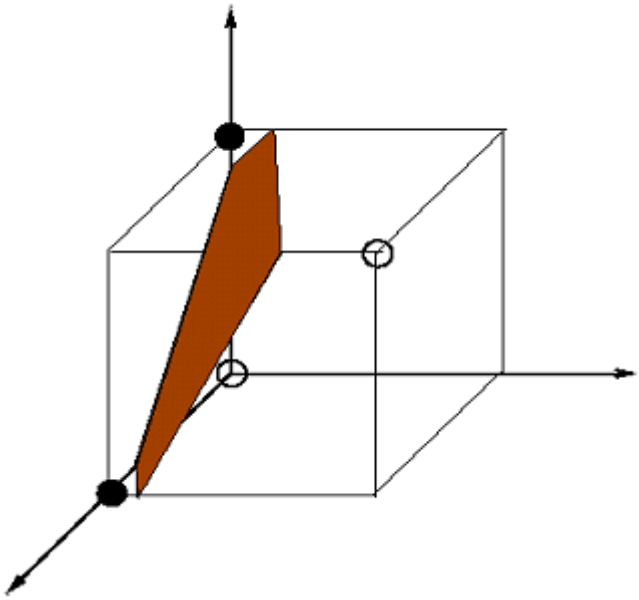
2.5 Illustration de transformation de cas non linéaire : le cas XOR

Le cas de XOR n'est pas linéairement séparable, si on place les points dans un plan à deux dimension, on obtient la figure suivante
Coordonnées des points : (0,0) ; (0,1) ; (1,0) ; (1,1)



Si on prend une fonction polynomiale (x , y) -> (x , y , x.y) qui fait passer d'un espace de dimension 2 à un espace de dimension 3, on obtient un problème en trois dimensions linéairement séparable :

(0,0) -> (0,0,0)
(0,1) -> (0,1,0)
(1,0) -> (1,0,0)
(1,1) -> (1,1,1)



3 Fondements mathématiques

Nous allons détailler dans les paragraphes ci-dessous les principas mathématiques sur lesquels repose SVM.

3.1 Problème d'apprentissage

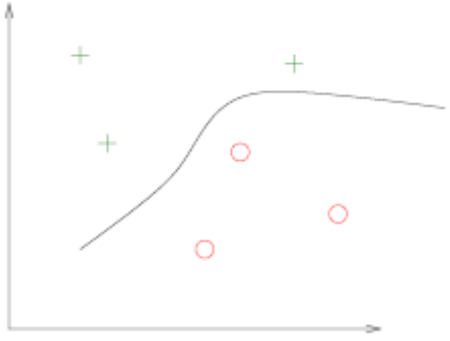
On s'intéresse à un phénomène f (éventuellement non déterministe) qui, à partir d'un certain jeu d'entrées x, produit une sortie y = f(x).
Le but est de retrouver cette fonction f à partir de la seule observation d'un certain nombre de couples entrée-sortie {(xi; yi) : i = 1, .. , n} afin de « prédire » d'autres évènements.

On considère un couple (X, Y) de variables aléatoires à valeurs dans $\boldsymbol{X}$ x $\boldsymbol{Y}$.
Seul le cas $\boldsymbol{Y}$ = {-1, 1} (classification) nous intéresse ici (on peut facilement étendre au cas card($\boldsymbol{Y}$) = m > 2 et au cas $\boldsymbol{Y} = \mathbb{R}$). La distribution jointe de (X, Y) est inconnue.

Sachant qu'on observe un échantillon S = {(X1, Y1),... ,(Xn, Yn)} de n copies indépendantes de (X, Y), on veut: construire une fonction h : $X \rightarrow Y$ telle que P(h(X) != Y) soit minimale.

Illustration :

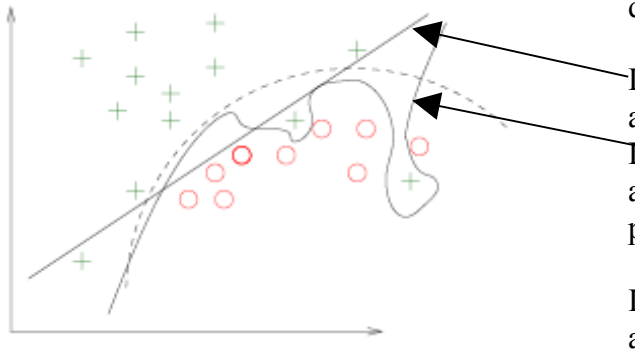
Trouver une frontière de décision qui sépare l'espace en deux régions (pas forcément connexes).



Connaissant h, on peut en déduire la classification des nouveaux points c'est à dire trouver une frontière de décision.

Le problème est de trouver une frontière assez éloignée des points de différentes classes. C'est ce qui constituera l'un des problèmes majeurs de classification grâce aux SVMs.

Sur et sous-apprentissage :



Si les données sont générées par un modèle quadratique :

Le modèle linéaire est en situation de sous-apprentissage
Le modèle de haut degré est en situation de sur-apprentissage (apprentissage par coeur)

Il faut donc trouver un compromis entre adéquation aux données et complexité pour pouvoir généraliser.

3.2 Classification à valeurs réelles

Plutôt que de construire directement h : $\boldsymbol{X} \rightarrow \{-1, 1\}$, on construit :
f : $\boldsymbol{X} \rightarrow \boldsymbol{R}$ (ensemble des réels).
La classe est donnée par le signe de f ;
h = signe(f) .

L'erreur se calcule avec P(h(X) != Y) = P(Yf(X) $\leq$0). Ceci donne une certaine idée de la confiance dans la classification. Idéalement, |Yf(X)| est proportionnel à P(Y|X).
Yf(X) représente la marge de f en (X,Y).
Le but à atteindre est la construction de f et donc h. Nous allons voir comment y parvenir.

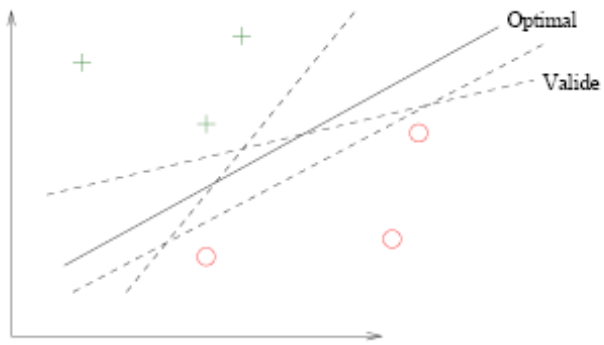
3.2.1 Transformation des entrées

Il est peut être nécéssaire de transformer les entrées dans le but de les traiter plus facilement.
$\boldsymbol{X}$ est un espace quelconque d'objets.
On transforme les entrées en vecteurs dans un espace $\boldsymbol{F}$ (feature space) par une fonction :
$\boldsymbol{\Phi} : \boldsymbol{X} \rightarrow \boldsymbol{F}$

$\boldsymbol{F}$ n'est pas nécessairement de dimension finie mais dispose d'un produit scalaire (espace de Hilbert). L'espace de Hilbert est une généralisation de l'espace euclidien qui peut avoir un nombre infini de dimensions.

La non-linéarité est traitée dans cette transformation, on peut donc choisir une séparation linéaire (on verra plus loin comment on arrive à ramener un problème non linéaire en un problème linéaire classique).

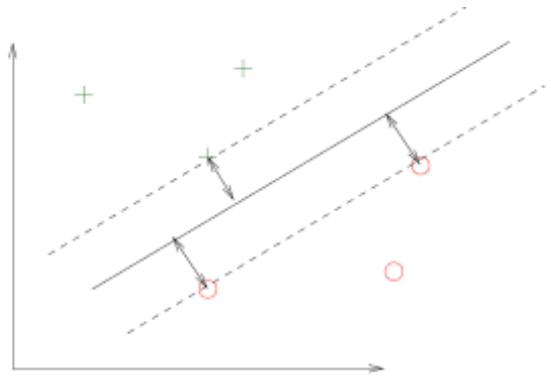
Dès lors, il s'agit de choisir l'hyperplan optimal qui classifie correctement les données (lorsque c'est possible) et qui se trouve le plus loin possible de tous les points à classer.



Mais l'hyperplan séparateur choisi devra avoir une marge maximale.

3.2.2 Maximisation de la marge

La marge est la distance du point le plus proche à l'hyperplan.



Dans un modèle linéaire (cf. figure ci-dessus), on a f(x) = w.x + b. L'hyperplan séparateur(frontière de décision) a donc pour équation w.x + b = 0.

La distance d'un point au plan est donnée par d(x) = |w.x + b|/||w||
L'hyperplan optimal est celui pour lequel la distance aux points les plus proches (**marge**) est maximale. Soient x1 et x2 eux points de classes différentes (f(x1) = +1 et f(x2) = -1)
($\boldsymbol{w}.\boldsymbol{x}1$) + b = +1 et ($\boldsymbol{w}.\boldsymbol{x}2$) + b = -1 donc ($\boldsymbol{w}.(\boldsymbol{x}1 - \boldsymbol{x}2)$) = 2
D'où : (w/||w|| . ($\boldsymbol{x}1 - \boldsymbol{x}2$)) = 2/||w||.
On peut donc en déduire que maximiser la marge revient à minimiser ||w|| sous certaines contraintes que nous verrons dans les paragraphes suivants.

3.2.3 Problème primal

Un point (x; y) est bien classé si et seulement si yf(x) > 0
Comme le couple (w, b) est défini à un coefficient multiplicatif près, on s'impose yf(x) ≥ 1
On en déduit (en s'appuyant également sur le paragraphe précédent), le problème de minimisation sous contraintes suivantes :

$$\begin{cases} \min \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \forall i,\ y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) \geq 1 \end{cases}$$

Il peut etre en effet plus aisé de minimiser ||w||² plutot que directement ||w||.

3.2.4 Problème dual

On passe du problème primal au problème dual en introduisant des multiplicateurs de Lagrange pour chaque contrainte.
Ici on a une contrainte par exemple d'apprentissage

$$\begin{cases} \max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \\ \forall i, \alpha_i \geq 0 \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

C'est un problème de programmation quadratique de dimension n (nombre d'exemples).
On défini ainsi la matrice suivante appelée **« matrice hessienne »** : $(\mathrm{Xi.Xj})_{i,j}$ qui représente la matrice des produits des entrées X (La notation matricielle permettant de résoudre plus facilement le problème en informatique).

On montre que si les α_i^* sont solutions de ce problème alors on a :

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \boldsymbol{x}_i$$

Seuls les α_i correspondant aux points les plus proches sont non-nuls. On parle de ***vecteurs de support***.
La fonction de décision associée est donc :

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i^* y_i \boldsymbol{x}_i \cdot \boldsymbol{x} + b$$

Il existe néanmoins des cas où on ne peut pas classer les entrées de façon linéaire.

3.3 La non linéarité (cas non séparable/ marge molle)

On part du problème primal linéaire et on introduit des variables « ressort » pour assouplir les contraintes.

$$\begin{cases} \min \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{n} \xi_i \\ \forall i,\ y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) \geq 1 - \xi_i \end{cases}$$

On pénalise par le dépassement de la contrainte.

On en déduit le problème dual qui a la même forme que dans le cas séparable:

$$\begin{cases} \max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \\ \forall i, 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

La seule différence est la borne supérieure C sur les α.

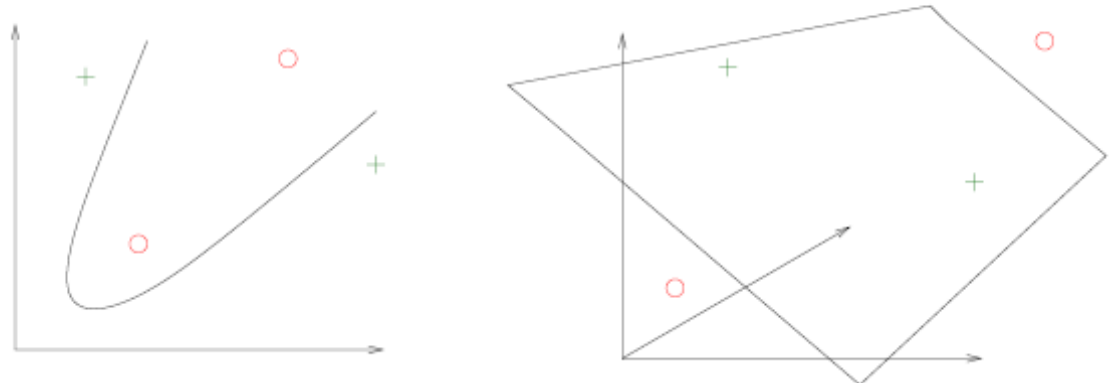
3.3.1 Fonction noyau (kern)

Dans le cas linéaire, on pouvait transformer les données dans un espace où la classification serait plus aisée. Dans ce cas, l'espace de redescription utilisé le plus souvent est ***R*** (ensemble des nombres réels). Il se trouve que pour des cas non linéaires, cet espace ne suffit pas pour classer les entrées. On passe donc dans un espace de grande dimension.

$$\begin{aligned}\Phi \ : \ & \mathbb{R}^d \rightarrow \mathcal{F} \\ & \boldsymbol{x} \mapsto \Phi(\boldsymbol{x})\end{aligned}$$

Avec card(***F***) > d.

Exemple :



Le passage dans $\mathcal{F} = \mathbb{R}^3$ rend possible la séparation linéaire des données.

On doit donc résoudre

$$\begin{cases} \max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j) \\ \forall i, 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

et la solution a la forme

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i^* y_i \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}) + b$$

Le problème et sa solution ne dépendent que du produit scalaire **Φ**(x).**Φ**(x').
Plutôt que de choisir la transformation non-linéaire **Φ** : ***X*→*F***, on choisit une fonction k :***X***x***X***→ ***R*** (nombres réels) appelée ***fonction noyau***.

Elle représente un produit scalaire dans l'espace de représentation intermédiaire. Du coup k est linéaire (ce qui nous permet de faire le rapprochement avec le cas linéaire des paragraphes précédents).
Cette fonction traduit donc la répartition des exemples dans cet espace k(x,x') = **Φ**(x).**Φ**(x').
Lorsque k est bien choisie, on n'a pas besoin de calculer la représentation des exemples dans cet espace pour calculer **Φ**.

Exemple :
Soit $\boldsymbol{x} = (x_1, x_2)$ et $\Phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$.
Dans l'espace intermédiaire, le produit scalaire donne

$$\begin{aligned}\Phi(\boldsymbol{x}) \cdot \Phi(\boldsymbol{x}') &= x_1^2 x_1'^2 + 2x_1x_2x_1'x_2' + x_2^2 x_2'^2 \\ &= (x_1x_1' + x_2x_2')^2 \\ &= (\boldsymbol{x} \cdot \boldsymbol{x}')^2\end{aligned}$$

On peut donc calculer **Φ**(x).**Φ**(x') sans calculer **Φ :** k(x,x')=(x.x')².
k représentera donc le noyau pour les entrées correspondantes mais devra néanmoins remplir certaines conditions.

3.3.2 Condition de Mercer

Une fonction k symétrique est un noyau si $(k(xi,xj))_{i,j}$ est une matrice définie positive.
(cf: http://www.techno-science.net/?onglet=glossaire&definition=5188).
Dans ce cas, il existe un espace ***F*** et une fonction **Φ** tels que k(x,x') = **Φ**(x).**Φ**(x').

Problèmes :

- Cette condition est très difficile à vérifier
- Elle donne pas d'indication pour la construction de noyaux
- Elle ne permet pas de savoir comment est **Φ**

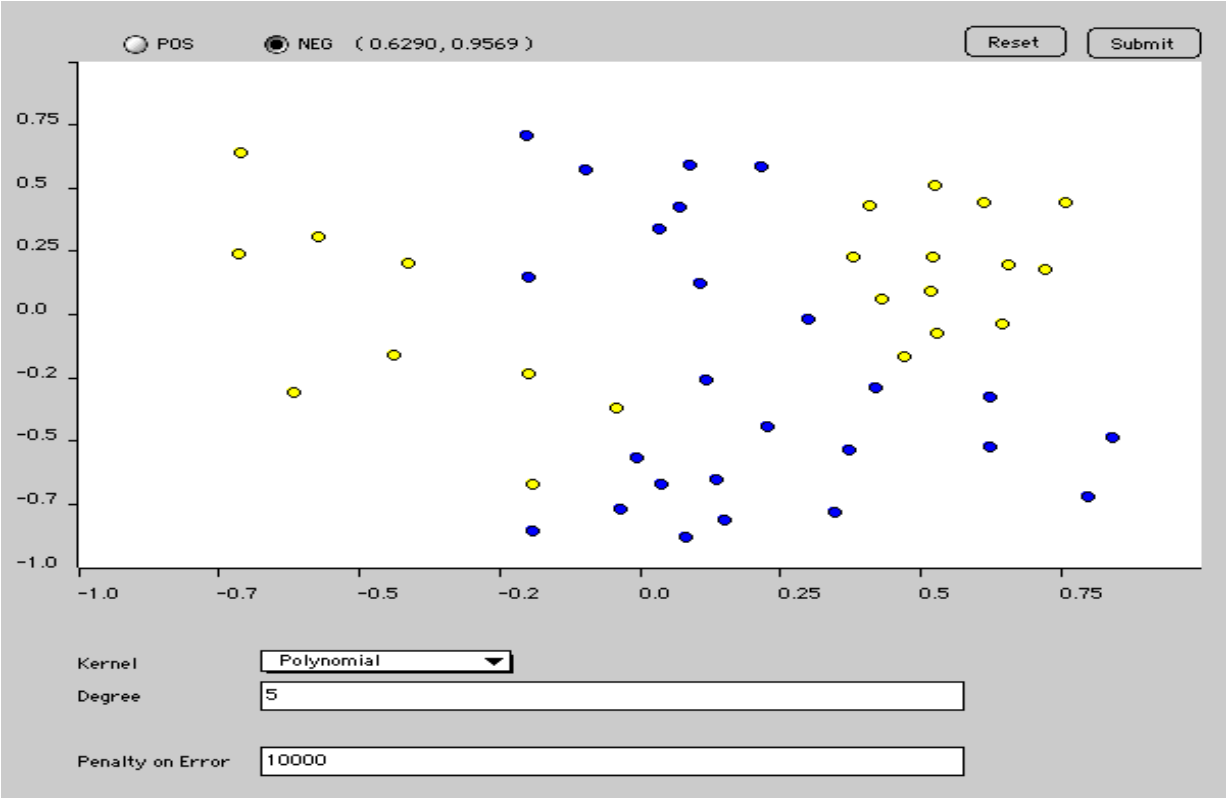
En pratique, on combine des noyaux simples pour en obtenir de plus complexes.

Exemples de noyaux :

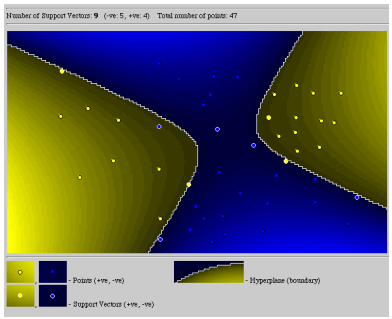
- Linéaire $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x} \cdot \boldsymbol{x}'$
- Polynomial $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} \cdot \boldsymbol{x}')^d$ ou $(c + \boldsymbol{x} \cdot \boldsymbol{x}')^d$
- Gaussien $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\|\boldsymbol{x}-\boldsymbol{x}'\|^2/\sigma}$
- Laplacien $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\|\boldsymbol{x}-\boldsymbol{x}'\|_1/\sigma}$

Petite étude comparative des noyaux polynomial et gaussien

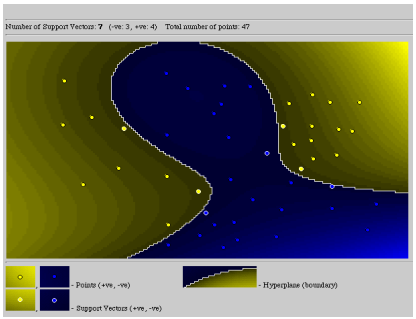
Soient les données d'apprentissage suivantes :



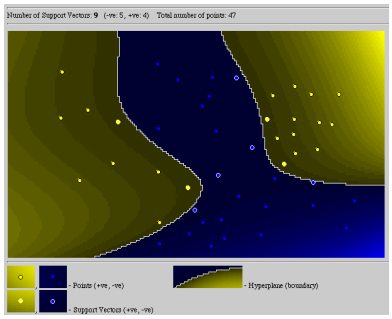
On réalise la simulation suivante en ajoutant des données à classer et en utilisant un noyau polynomial et un noyau gaussien. On fixe dans les deux cas la constante C à 10000
La distribution initiale est la suivante : 47 données d'apprentissage (22+ et 25 -). En bleu les données de classe + et en jaune celles de classe -.



(5-, 4+)

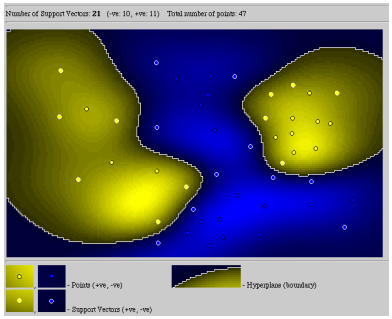


(3-, 4+)

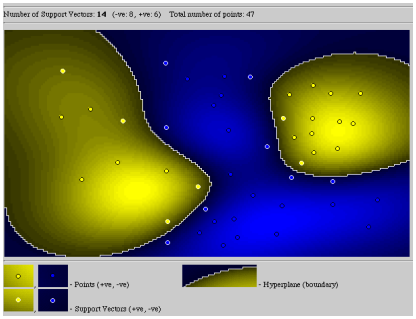


(5-, 4+)

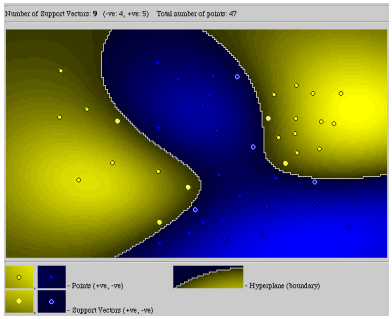
Noyau polynomial de degré : 2, 5 et 8



(10-, 11+)



(8-, 6+)



(4-, 5+)

Noyau gaussien σ = 2, 5, 10

On remarque en général le noyau gaussien donne de meilleurs résultats et groupe les données dans des paquets nets.

3.4 Temps de calcul et convergence

- **Complexité**

Nous allons évaluer la complexité(temps de calcul) de l'algorithme SVM.
Elle ne dépend que du nombre des entrées à classer (d) et du nombre de données d'apprentissage (n).

On montre que cette complexité est polynomiale en n.

$$dn^2 \leq \text{Complexite} \leq dn^3$$

Taille de la matrice hessienne = n²
En effet, on doit au moins parcourir tous les éléments de la matrice ainsi que toutes les entrées..
Pour un très grand nombre de données d'apprentissage, le temps de calcul explose. C'est pourquoi les SVMs sont pratiques pour des « petits » problèmes de classification.

- **Pourquoi SVM marche ?**

Les noyaux précédents qui sont les plus utilisés, remplissent les conditions de mercer (facile à vérifier une fois qu'on a le noyau).

Normalement, la classe (le nombre) des hyperplans de $\mathbb{R}^d$ est de $d\mathrm{H} = d + 1$.
Mais la classe des hyperplans de marge $1/\|w\|$ tels que $\|w\|^2 \leq c$ est bornée par : $d\mathrm{H} \leq \mathrm{Min}\,(R^2 c, d) + 1$
Où R est le rayon de la plus petite sphère englobant l'échantillon d'apprentissage S
Donc dH peut être beaucoup plus petit que la dimension d de l'espace d'entrée $\boldsymbol{X}$; il est donc toujours possible d'en trouver un c'est la raison poue laquelle.

4 Les domaines d'applications

SVM est une méthode de classification qui montre de bonnes performances dans la résolution de problèmes variés. Cette méthode a montré son efficacité dans de nombreux domaines d'applications tels que le traitement d'image, la catégorisation de textes ou le diagnostiques médicales et ce même sur des ensembles de données de très grandes dimensions

La réalisation d'un programme d'apprentissage par SVM se ramène à résoudre un problème d'optimisation impliquant un système de résolution dans un espace de dimension conséquente. L'utilisation de ces programmes revient surtout à sélectionner une bonne famille de fonctions noyau et à régler les paramètres de ces fonctions . Ces choix sont le plus souvent faits par une technique de validation croisée, dans laquelle on estime la performance du système en la mesurant sur des exemples n'ayant pas été utilisés en cours d'apprentissage. L'idée est de chercher les paramètres permettant d'obtenir la performance maximale. Si la mise en oeuvre d'un algorithme de SVM est en général peu coûteuse en temps, il faut cependant compter que la recherche des meilleurs paramètres peut requérir des phases de test assez longues.

5 SVM dans Oracle (ODM : Oracle Data Mining)

Comme on l'a vu précédemment, SVM est une technologie assez performante pour classifier un ensemble de points et donc pour faire du datamining. De plus SVM peut être appliqué à plusieurs domaines (bioinformatique pour la reconnaissance de gènes, etc..). L'introduction des SVM dans les années 1990 dans certaines applications a érigé cette technique de classification au rang d'outil standard pour l'apprentissage et le datamining.
Ce sont les raisons pour lesquelles Oracle l'a intégré dans sa version 10g.

5.1 Spécificités de SVM dans oracle 10g

Les fonctionnalités suivantes ont été ajoutées à SVM dans oracle 10g :

5.1.1 Détection d'anomalies (Anomaly detection)

Elle consiste en l'identification d'échantillons anormaux. Un modèle de détection d'anomalies prédit si un point convient à la distribution ou non. Si ce n'est pas le cas, ce point peut être défini comme un outlier ou juste comme le point d'une classe qui n'a pas encore été déterminée.
La détection d'anomalie est une fonction de mining dans l' « oracle data miner interface » et un modèle de classification dans les interfaces ODM java et PL/SQL.

5.1.2 L'apprentissage actif (Active learning)

Les modèles d'apprentissage actif augmentent en même temps que la taille des données . Cette propriété fait qu'on va se limiter à relativement peu de données d'entraînements (100.000 au maximum).
Le critère ici est la limite du nombre des vecteurs de support. Une fois cette limite atteinte, la construction s'arrête. En fait ceci est pertinent dans la mesure où au delà de cette limite, les performances sont quasi inchangées.
L'apprentissage actif force l'algorithme SVM à se restreindre aux données qui apportent plus d'information (les plus intéressantes).
Cette option peut être appliquée à tous les modèle SVM (classification, régression et « one-class ») et peut être désactivée.

5.1.3 Echantillonage et choix du noyau

Pour la classification, SVM fait des couches d'échantillons durant la construction du modèle. L'algorithme parcourt l'ensemble des données construites et sélectionne un échantillon qui est réglé par rapport aux données cibles.
Oracle 10g implémente deux type de noyaux : linéaire et gaussien. Le choix du noyau est faite de façon automatique ou manuelle.

5.2 Le package DBMS_DATA_MINING et SVM

DBMS_DATA_MINING est le package dédié au datamining pour des applications Pl/sql.
Le tableau ci-dessous décrits les variables nécessaires à l'exécution de l'algorithme SVM.

Nom	Description
svms_kernel_function	Type de noyau : • svms_linear (pour noyau linéaire) • svms_gaussian (pour noyau gaussien) La valeur par défaut est : svms_linear
svms_kernel_cache_size	TO_CHAR(*numeric_expr* > 0) Valeur de la taille du cache pour l'exécution de l'algorithme. Elle concerne uniquement les noyaux gaussiens. La valeur par défaut est : 50000000 bytes
svms_conv_tolerance	TO_CHAR(*numeric_expr* > 0) Tolérance de convergence pour l'algorithme La valeur par défaut est : 0.001.
svms_std_dev	TO_CHAR(*numeric_expr* > 0) Valeur de la deviation standard. Conerne uniquement les noyaux gaussiens. La valeur par défaut est estimée par l'algo en fonction des données.
svms_complexity_factor	TO_CHAR(*numeric_expr* > 0) Valeur du facteur de complexité . La valeur par défaut est estimée par l'algo en fonction des données.
svms_epsilon	TO_CHAR(*numeric_expr* > 0) Valeur du facteur epsilon pour l'algorithme de régression par SVM. La valeur par défaut est estimée par l'algo en fonction des données.

5.3 Arbres de décision vs SVM

Nous allons ici présenter une étude comparative entre l'algorithme de classification SVM (avec un noyau gaussien) et celui des arbres de décision (A.D) tous deux implémentés par oracle sur deux cas pratiques.
Cette étude a été réalisée par le département informatique des sciences et statistiques de l'université de Rhode Island.

5.3.1 Cancer du poumon dans le Wisconsin

Ce test a été réalisé sur 645 enregistrements de patients dont 512 ont été utilisées comme données d'entraînement et les 133 autres comme données test (données à prédire).
Les attributs sont définis de la manière suivante :

Sample code number: id number
Clump Thickness: 1, 2,3,4,5,6,7,8,9,10.
Uniformity of Cell Size: 1, 2,3,4,5,6,7,8,9,10.
Uniformity of Cell Shape: 1, 2,3,4,5,6,7,8,9,10.
Marginal Adhesion: 1, 2,3,4,5,6,7,8,9,10.
Single Epithelial Cell Size: 1, 2,3,4,5,6,7,8,9,10.
Bare Nuclei: 1, 2,3,4,5,6,7,8,9,10.
Bland Chromatin: 1, 2,3,4,5,6,7,8,9,10.
Normal Nucleoli: 1, 2,3,4,5,6,7,8,9,10.
Mitose: 1,2,3,4,5,6,7,8,9,10.

On cherche donc à savoir ici si en fonction de ces paramètres (attributs) une tumeur est bénigne ou maligne. L'attribut de prédiction est le suivant :

Class: 2-benign, 4-malignant
Les données sont réparties de la façcon suivante : 65% benign et 35 % malignant

Ils obtiennent les matrices de confusion suivantes :

Algorithme de SVM utilisant un noyau gaussien

		Predicted	
		2 (B)	4 (M)
Actual	2 (B)	94	1
	4 (M)	0	39

Précision de 99,3%

Algorithme de A.D

		Predicted	
		2 (B)	4 (M)
Actual	2 (B)	86	9
	4 (M)	1	38

Précision de 92,5%

Remarque :

Voici la configuration initiale des paramètres nécessaires à l'exécution de l'algorithme SVM

SVMS_CONV_TOLERANCE = .001
SVMS_KERNEL_CACHE_SIZE = 50000000
SVMS_STD_DEV = 3.7416573867739413
SVMS_COMPLEXITY_FACTOR = 1.1959376673823801

5.3.2 Base de données pour les spams

Chaque tuple représente un email qui peut être considéré comme un Spam ou pas. Les 57 attributs continus de ces tuples décrivent des fréquences des mot et des caractères dans les messages. Ils ont enregistré 4601 tuples dont 39% sont des spams et 61% des e-mails normaux.
C'est un problème binaire de classification où tous les attributs indépendants sont continus. Ils ont pris 3520 tuples pour l'apprentissage et 811 pour les tests.

Ils obtiennent les matrices de confusion suivantes :

Algorithme de SVM utilisant un noyau gaussien

		Predicted	
		0	1
Actual	0	522	20
	1	15	254

Précision de 95,3%

Algorithme de A.D

		Predicted	
		0	1
Actual	0	441	33
	1	28	309

Précision de 92,5%

Remarque :

Voici la configuration initiale des paramètres nécessaires à l'exécution de l'algorithme SVM :

SVMS_CONV_TOLERANCE = .001
SVMS_KERNEL_CACHE_SIZE = 50000000
SVMS_STD_DEV = 4.812661641473027
SVMS_COMPLEXITY_FACTOR = .75904342468903196

A l'issue de ces deux exemples, on peut en déduire que le modèle de SVM donne de meilleurs résultats que celui des arbres de décision. Ce qui s'explique en grande partie par la puissance du modèle mathématique utilisé par SVM

6 Conclusion

Dans cet article, nous avons tenté de présenter de manière simple et complète le concept de système d'apprentissage introduit par Vladimir Vapnik, les « Support Vector Machine » Nous avons donné une vision générale et une vision purement mathématiques des SVM. Cette méthode de classification est basé sur la recherche d'un hyperplan qui permet de séparer au mieux des ensembles de données. Nous avons exposé les cas linéairement séparable et les cas non linéairement séparables qui nécessitent l'utilisation de fonction noyau (kernel) pour changer d'espace. Cette méthode est applicable pour des tâches de classification à deux classes, mais il existe des extensions pour la classification multi classe.

Nous nous somme ensuite intéressé aux différents domaines d'application et nous avons insisté sur l'utilisation des SVM dans Oracle (ODM). Les logiciels de SVM sont disponibles sur Internet et les expériences sont faciles à réaliser. On obtient souvent en les utilisant des résultats comparables à ceux obtenus avec d'autres techniques et les paramètres à régler sont moins nombreux.

Il existe des extensions que nous n'avons pas présenté, parmi lesquelles l'utilisation des SVM pour des tâches de régression, c'est-à-dire de prédiction d'une variable continue en fonction d'autres variables, comme c'est le cas par exemple dans de la prédiction de consommation électrique en fonction de la période de l'année, de la température, etc. Le champ d'application des SVM est donc large et représente une méthode de classification intéressante.

7 Références

www.kernel-machines.org
www.kernel-methods.net
www.support-vector.net

- Introduction aux « Supports Vector Machines » (SVM),Olivier Bousquet, Ecole Polytechnique, Palaiseau
- SVM, un méthode de classification binaire par apprentissage, Millet Christophe
- Support Vector Machines and other kernel-based learning methods, Cristianini & Shawe-Taylor, Université de Cambridge
-The nature of statistical learning, Vapnik
*- SVM in Oracle Database 10g: Removing the Barriers toWidespread Adoption of Support Vector Machines***,** *Boriana L. Milenova*