TP/TD Optimisation Classique 2020 - **Majeure Science des Données**

Séparateurs à Vaste Marge pour la classification (et la régression)

durée ~ 3heures + travail personnel

Travail à réaliser par groupes de 3 ou 4. A rendre sur Campus: un fichier compressé contenant le rapport (6 à 8 pages) et votre code .
***La partie régression (SVR) du TP n'est pas à réaliser** (faute de temps) . Elle est cependant laissée dans l'énoncé pour les élèves qui pourraient être intéressés par cette méthode (voir alors le fichier « présentation SVR » sur Campus)*

Objectifs / Plan de travail:

Ce TP/TD, comme le cours qui le précède, est à la croisée des chemins entre méthodes d'optimisation continue et science des données. Certaines parties seront traitées soit à la fin du cours, soit sous forme de TD pendant les séances de TP.

Il s'agit de mettre en œuvre des méthodes d'optimisation contrainte, et plus particulièrement les méthodes numériques duales, pour *construire vous-même* :

- un outil de *classification automatique*: une "Machine à Vecteur Support" (SVM).
- (un outil de *régression* très proche du premier dans son principe et sa réalisation – il faudra modifier le premier pour obtenir le second (SVR).)

Il faudra dans chaque cas tester l'efficacité de ces outils sur différents jeux de données.

La programmation sera faite à l'aide de Matlab.

Pour les SVM et SVR, la trame du plan de travail est la même. On vous aide plus pour les SVM et on vous laisse plus d'autonomie pour les SVR.

Etape 1: Comprendre la finalité et le principe des SVM (et SVR), *avant de venir en TP (sera abordé en cours et voir aussi la documentation fournie – deux articles).* Notamment, avoir compris comment chacune des deux méthodes se traduit par la résolution d'un problème d'optimisation contraint, formulations primale et duale. Avoir compris comment on passe du linéaire au non-linéaire grâce à l'utilisation de noyaux (*kernel trick*). Avoir compris le principe des marges souples. Vous pourrez poser des questions sur ces points lors du TP. Ce document traite dans une première partie assez longue les aspects « théoriques » , parfois sous forme de TD (questions qui vous sont posées).

Etape 2:En TP Comprendre l'architecture du logiciel à construire, notamment les structures de données d'entrée, et les fonctions noyau et produit scalaire. Vous aurez à développer vous-mêmes le cœur du logiciel : la mise en œuvre de la méthode d'Uzawa pour chacun des deux problèmes, qui sont très proches. Ecrire l'algorithme de la méthode d'Uzawa (papier) en traitant les contraintes inégalité par la méthode de projection.

Etape 3: Programmer avec Matlab la méthode d'Uzawa, tester sur des cas simples au début. Etudier la convergence de l'algorithme en fonction de ses paramètres. Modifier progressivement le code pour traiter marges souples et cas non linéaire (avec noyau). Faire ce travail d'abord pour les SVM (3hrs) puis pour les SVR (2hrs) par modification de code

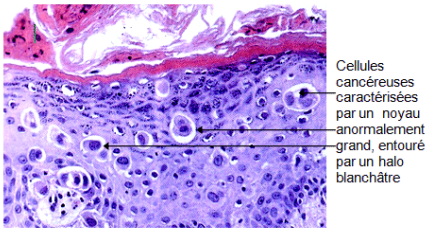
Etape 4 : Enfin, utiliser les SVR pour générer un « modèle » sur le cas « fil rouge » des antennes de la Loire. Si le temps le permet, lancer un algorithme d'optimisation locale pour positionner les antennes à partir du modèle (*surrogate* optimisation)

Vous êtes aidés et guidés par le document ci-après

Partie 1 : Les SVM pour la classification

1- Les bases de la méthode (à comprendre avant le TP)

Document : Machines à vecteurs de support – une introduction , Dominik Francoeur
Introduction sur un exemple



Epiderme de glande mammaire envahi par des cellules cancéreuses

Exemple : on prend, à l'aide d'un microscope couplé à une caméra N&B, n mesures sur une cellule : taille du noyau, taille du cytoplasme, niveau de gris du cytoplasme, épaisseur de la membrane …. etc
La cellule est alors représentée par un point de R^n , de coordonnées la valeur des mesures (paramètres de la cellule).

On voudrait déterminer automatiquement (sans intervention 'humaine') si la cellule est cancéreuse ou pas, sachant que cette pathologie peut être caractérisées par les (ou certains) paramètres mesurés.

Méthode : on dispose d'un échantillon de p cellules, donc p points { x_i } de R^n , saines et cancéreuses, dont on sait celles qui sont saines et celles qui sont cancéreuses. On affecte à chaque point xi un « label » li qui vaut 1 ou -1, suivant que la cellule est cancéreuse ou pas. Cet échantillon est appelé « échantillon d'apprentissage ». Il va servir à construire une hypersurface qui *sépare au mieux* (idéalement) les cellules saines des cellules cancéreuses. Cette hypersurfarce est vue comme la surface de niveau 0 d'une fonction de R^n dans R appelée fonction de décision. En général, la situation n'est pas si idéale, des cellules saines pouvant se trouver dans la zone des cellules cancéreuses, ou/et vice versa (*outliers*)

Le problème étant de déterminer l'hypersurface de démarcation entre les deux régions. Cette hypersurface peut être vue comme l'isovaleur d'une fonction f : Rn → R . (f(x)=0). On appelle f : *fonction de décision*. C'est le rôle des Séparateurs à Vaste Marge (ou Support Vector Machines)

L'hypersurface la plus simple que l'on puisse déterminer est l'hyperplan (SVM linéaire),
la fonction de décision est alors: $f(x) = w^T x + b$,

$f(x) = w^T x + b \geq 0$: la cellule x est saine

$f(x) = w^T x + b \leq 0$: la cellule x est cancéreuse

On a ainsi partitionné R^n à l'aide de l'hypersurface f(x)=0. Lorsque l'on dispose d'une *nouvelle* cellule et de ses paramètres : $\tilde{x}$, le signe de $f(\tilde{x})$ détermine 'automatiquement' si la cellule est saine ou pas.

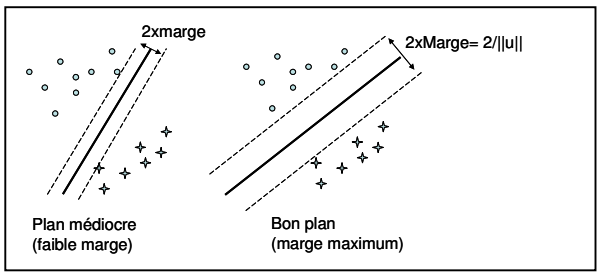
Wikipédia:

Les machines à vecteurs de support ou séparateurs à vaste marge (en anglais Support Vector Machine, SVM) sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination et de régression. Les SVM sont une généralisation des classifieurs linéaires.
Les SVM ont été développés dans les années 1990 à partir des considérations théoriques de Vladimir Vapnik sur le développement d'une théorie statistique de l'apprentissage : la théorie de Vapnik-Chervonenkis. Les SVM ont rapidement été adoptés pour leur capacité à travailler avec des données de grandes dimensions, le faible nombre d'hyper paramètres, leurs garanties théoriques, et leurs bons résultats en pratique.
Les SVM ont été appliqués à de très nombreux domaines (bio-informatique, recherche d'information, vision par ordinateur, finance ...). Selon les données, la performance des machines à vecteurs de support est de même ordre, ou même supérieure, à celle d'un réseau de neurones ou d'un modèle de mélanges gaussiens.

Analyse et position du problème d'optimisation pour le cas linéaire simple (pas d'outliers)

On considère un ensemble de p points (x_k) dans de R^n (points d'apprentissage). Ces points se divisent en deux familles, leur appartenance à l'une où l'autre des familles est marquée par un label l_k attaché à chaque point, qui vaut 1 pour l'une des familles et -1 pour l'autre.

On cherche à séparer (automatiquement) au mieux les deux familles par un hyperplan d'équation : ${}^T wx+b=0,\ \ w\in R^n, b\in R$. Et on suppose que cela est possible dans ce premier cas simple.



Les inconnues sont à priori w et b, mais notez qu'à ce stade, il n'y a pas unicité du couple (w,b) pour définir l'hyperplan. Le problème semble mal posé.

L'hyperplan optimum devra maximiser la *marge*, c'est-à-dire la demi-largeur de la bande symétrique de largeur maximale autour de l'hyperplan (voir dessin) qui ne contient aucun point.

- La distance d'un point x_k à l'hyperplan vaut $\left|\frac{{}^T wx_k+b}{\|w\|}\right|$ ou encore $\frac{l_k\left({}^T wx_k+b\right)}{\|w\|}$ si on oriente correctement w.
- Les points vérifiant $l_k\left({}^T wx_k+b\right)\geq 1$ sont situés hors d'une marge de demi-largeur $\frac{1}{\|w\|}$, que l'on voudra donc maximiser.
- Exprimer le problème comme un problème de minimisation quadratique avec contraintes linéaires, dont l'inconnue est le couple (w,b). Remarquer que b n'intervient pas dans la fonction coût, mais dans les contraintes. Ecrire le Lagrangien L(w,b,α) où α est le vecteur des multiplicateurs de Kuhn et Tucker, $\alpha\in R^p$. Après avoir calculé le gradient en w et en b de L, exprimer la fonction duale H(α). Est-elle bien concave ?
- Montrer qu'il s'agit alors de résoudre le problème dual, d'inconnue α (en changeant le signe de la fonction duale à maximiser pour obtenir un problème de minimisation standard):

$$\underset{\alpha\geq 0}{Min}\left(-H(\alpha)\right)=\underset{\alpha\geq 0}{Min}\left(\frac{1}{2}\sum_{i,j=1,p}\alpha_i\alpha_j l_i l_j x_i{}^T x_j-\sum_{i=1,p}\alpha_i\right)$$

que l'on écrira sous forme matricielle

$$\sum_{i=1,p} l_i\alpha_i=0,\qquad \alpha_i\geq 0$$

La matrice de la forme quadratique est-elle définie positive ? Conséquence : quel est le gradient de H(α) ? Remarquer que dans ce problème, les points xi n'interviennent que par leur produit scalaire deux à deux. On verra par la suite l'importance de cette remarque.

Aide : développement des calculs :

Problème primal : $Min\ \frac{1}{2}\|w\|^2 \qquad \forall k=1\ldots p, l_k\left({}^T wx_k+b\right)\geq 1$

Le lagrangien s'écrit : $L(w,b,\alpha)=\frac{1}{2}\|w\|^2+\sum_1^p\alpha_k\left(1-l_k\left({}^T wx_k+b\right)\right)$

On peut alors écrire la fonction duale : $H(\alpha) = \underset{w,b}{Min} L(w,b,\alpha)$. L est convexe en w,b (au sens large car affine dans la direction b).Pour chaque valeur de α Le minimum annule le gradient suivant (w ,b):
On peut noter w_α et b_α le couple réalisant le minimum

$$\nabla_w L(w_\alpha, b_\alpha, \alpha) = w_\alpha - \sum_1^p l_k \alpha_k x_k = 0, \qquad \nabla_b L(w_\alpha, b_\alpha, \alpha) = \sum_1^p l_k \alpha_k = 0$$

On remarque l'expression de w_α comme combinaison linéaire des x_k : $w_\alpha = \sum_1^p l_k \alpha_k x_k$

On remplace alors w par $w_\alpha = \sum_1^p l_k \alpha_k x_k$ dans L, de façon à obtenir l'expression de H(α) :

$$H(\alpha) = \sum_1^p \alpha_k + \frac{1}{2}\|w_\alpha\|^2 - \sum_1^p \alpha_k \left(l_k \left(\sum_{i=1}^p \alpha_i x_i \right)^T x_k \right) - b_\alpha \sum_1^p \alpha_k l_k$$

$$H(\alpha) = \sum_1^p \alpha_k - \frac{1}{2}\sum_{i,j}^p \alpha_i \alpha_j l_i l_j x_i^T x_j \qquad \text{en utilisant : } \sum_1^p \alpha_k l_k = 0$$

Pb dual : $\begin{cases} \underset{\alpha \geq 0}{Max}\left(H(\alpha)\right) \\ \sum_1^p \alpha_k l_k = 0 \\ \alpha_k \geq 0 \end{cases}$ on peut écrire $H(\alpha) = -\frac{1}{2}\alpha^T A \alpha + u^T \alpha$, avec A matrice pxp,

u vecteur de R^p constitués de 1

Il s'agit d'un problème quadratique avec contraintes linéaires, relativement simple.

Pour se ramener à une expression déjà vue, on peut écrire $H(\alpha) = -\frac{1}{2}\alpha^T A \alpha + u^T \alpha$, avec A matrice pxp, u vecteur de 1

Avec $A_{ij} = l_i l_j x_i^T x_j$, matrice définie positive, on remarque que les points interviennent à travers leur produit scalaire deux à deux.

Le gradient de H(α) s'exprime simplement :
$\nabla H(\alpha) = -A\alpha + u$

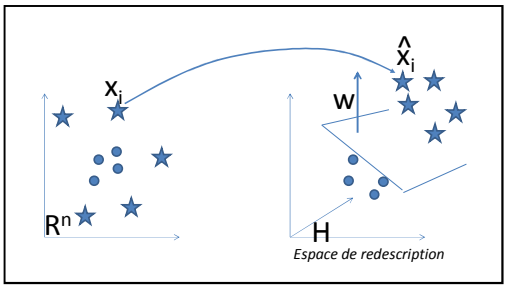
Une fois le vecteur des multiplicateurs de Lagrange trouvé par optimisation contrainte, il faut déterminer la constante b. Pour cela on se sert des points supports. Essayer de trouver par vous-mêmes comment.

$$f(x) = \langle w, x \rangle + b = \sum_{k=1\ldots p} \alpha_k l_k \langle x_k, x \rangle$$

Puis in fine on obtient la fonction de décision. Pour l'instant c'est une fonction affine de R^n dans R, dont la ligne de niveau 0 détermine l'hyperplan de R^n séparateur, et les lignes de niveau + et – 1 déterminent les marges.

Cas non linéaire, noyaux :

Dans le cas où les deux populations ne sont pas séparables par un hyperplan, l'idée consiste à plonger les x_i dans un espace de dimension plus grand (dit espace de re-description) où là, ils pourront être séparables par un hyperplan. En pratique, cet espace sera un espace fonctionnel H, de dimension infinie, un espace de Hilbert dans lequel le produit scalaire est défini à partir d'un « noyau ». On verra que pratiquement, cette idée est très facilement implémentable dans un code de SVM linéaire, car l'expression de la matrice A ne dépend que des produits scalaires $x_i x_j$



Noyau : une application symétrique semi-définie positive $K : R^n \times R^n \to R$

$K(x, y) = K(y, x)$

$K(x, x) \geq 0$

$\forall x_1 .. x_p$, p vecteurs de R^n, la matrice de Gram $G_{ij} = K(x_i, x_j)$ est définie positive

Exemple de noyau : noyau gaussien $K(x, y) = \exp(-\frac{\|x-y\|^2}{\sigma^2})$

On considère l'application $\Phi \quad x \in R^n \to \left(y \to K(x, y) \right)$ on peut aussi noter $x \to K(x,.)$ ou $x \to K_x$ ou encore plus simplement : $x \to \hat{x}$ (on conservera cette dernière notation , la plus légère)

On associe donc à chaque vecteur x de R^n, une fonction $\hat{x}$ de R^n dans R bâtie à partir du noyau positif K

On va construire un espace de Hilbert généré par les fonctions $\hat{x}$

On considère l'espace vectoriel engendré par les $\hat{x}$, x dans R^n : $V = Vect\{\hat{x}\}_{x \in R^n}$, ensemble des combinaisons linéaires *finies* de $\hat{x}$ $\quad f = \sum_1^r f_i \hat{x}_i$

Sur V, on définit le produit scalaire $\langle \hat{x}, \hat{y} \rangle = K(x, y)$ pour les fonctions génératrices

Et donc : $\langle f, g \rangle = \sum_{j=1}^{q} \sum_{i=1}^{r} f_i g_j \langle \hat{x}_i, \hat{x}_j \rangle$

Bien que muni d'un produit scalaire, l'espace vectoriel V n'est à priori pas un espace de Hilbert car non complet. Il faut le compléter en lui adjoignant les limites de suites de Cauchy, et étendre la définition du produit scalaire à ces limites. L'espace ainsi obtenu est noté H, espace des limites de suites de Cauchy de vecteurs de V.

On définit simplement le produit scalaire : si f_n et g_n sont deux suites de Cauchy dans V

$$f = \lim_{n\to\infty} f_n, \quad g = \lim_{n\to\infty} g_n \qquad \langle f, g\rangle_H = \lim_{n\to\infty} \langle f_n, g_n\rangle_H$$

De ce produit scalaire dérive une norme :
$\|f\|_H^2 = \langle f, f\rangle_H$

En particulier, pour les points de H images des points x de R^n
$\|\hat{x}\|_H^2 = K(x,x)$ et aussi bien sûr : $\langle \hat{x}, \hat{y}\rangle_H = K(x,y)$

On dispose dans R^n de p points x_i que l'on labellise comme précédemment avec l_i.
On « envoie » ces points dans H par l'application Φ :

On a donc p « points » de H : $\{\hat{x}_i\}_{i=1\ldots p}$
On cherche dans H le meilleur hyperplan séparateur. Le calcul sera formellement pratiquement le même que précédemment

Dans H, l'équation d'un hyperplan et l'expression de la distance à cette hyperplan est la même que dans R^n. Donc l'expression du problème primal est la même :

On cherche le couple $(w,b) \in H \times R$ qui réalise l'hyperplan de marge maximale séparant les points $\{\hat{x}_i\}_{i=1\ldots p}$

Problème primal **:**

$$Min \frac{1}{2}\|w\|^2 \qquad \forall k = 1\ldots p, l_k\left(\langle w, \hat{x}_k\rangle_H + b\right) \geq 1$$

On écrit le Lagrangien, la fonction duale…. En faisant bien attention d'écrire produits scalaires et normes dans H. Produit scalaire qui, pour les images de points de R_n uniquement, s'écrit simplement avec le noyeau K.

On a comme précédemment :

$w_\alpha = \sum_1^p l_k \alpha_k \hat{x}_k = 0$, égalité dans H

Mais attention : w est ici un vecteur de l'espace de redescription H et pas de R^n ! On n'a pas accès à w comme dans le cas linéaire, mais ce n'est pas important puisque ce qui nous intéresse est la fonction de décision, qui ne dépend que du produit scalaire de w avec les vecteurs $\hat{x}_k$, et on connait ces produites scalaires à partir du noyau puisque $\langle \hat{x}, \hat{x}_i\rangle = K(x, x_i)$

$$H(\alpha) = \sum_1^p \alpha_k - \frac{1}{2}\sum_{i,j}^p \alpha_i \alpha_j l_i l_j K(x_i, x_j)$$

Problème dual : $\begin{cases} Max\left(H(\alpha)\right) \\ \sum_1^p \alpha_k l_k = 0 \quad \alpha_k \geq 0 \end{cases}$

L'algorithme d'Uzawa se déroule de la même façon, avec la matrice $A_{ij} = l_i l_j K(x_i, x_j)$

(la projection sur le plan $\sum_{k=1\ldots p} \alpha_k l_k$ se fait avec le produit scalaire euclidien de R^p ....)

On calcule b de la même façon.

Etant donné un nouveau point x de R^n , la fonction de décision s'écrit

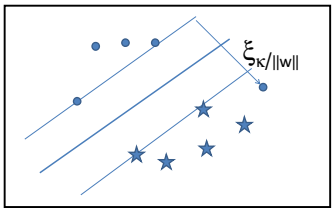
$$f(x) = \left\langle w^*, \hat{x} \right\rangle_H + b^* = \left(\sum_1^p l_k \alpha_k^* K(x, x_k) \right) + b^*$$

La fonction de décision de l'algorithme dans R^n étant exprimée uniquement en fonction des produits scalaires ${}^T x_i x_j$, la modification de l'algorithme est minimale et s'obtient en remplaçant le produit scalaire ${}^T x_i x_j$ dans R^n par le produit scalaire $K(x_i, x_j) = \left\langle \Phi(x_i), \Phi(x_j) \right\rangle_H$ des vecteurs images de vecteurs de R^n dans H. (K=noyau).

SVM à marge souple (cas non séparable)

On introduit des variables d'écart *positives* pour chaque point : $\{\xi_i\}_{i=1\ldots p}$. On permet cette fois à chaque point d'être dans une marge « assouplie » grâce à sa variable d'écart . Plus la variable d'écart est grande, moins bien est classé le point.

$$\forall k = 1\ldots p, \qquad l_k \left(\left\langle w, \hat{x}_k \right\rangle + b \right) \geq 1 - \xi_i$$



On veut alors maximiser la marge tout en minimisant les variables d'écart. On « pénalise » le problème d'optimisation en ajoutant le terme (positif) $C \sum_{k=1..p} \xi_k$

le coefficient C règle le compromis entre la taille de la marge et le nombre de données mal classées. Si C est grand, il y aura peu de données mal classées mais la marge sera petite…

$$Min \frac{1}{2}\|w\|^2 + C \sum \xi_i \qquad \forall k = 1\ldots p, l_k \left(\left\langle w, \hat{x}_k \right\rangle + b \right) \geq 1 - \xi_k$$
$$-\xi_i \leq 0$$

Lagrangien et problème dual :

$$L(w, b, \xi) = \frac{1}{2}\|w\|^2 + C \sum_{k=1..p} \xi_k \quad + \sum_{k=1..p} \alpha_k \left(1 - \xi_k - l_k \left(\left\langle w, \hat{x}_k \right\rangle + b \right) \right) - \sum_{k=1..p} \beta_k \xi_k$$

$$\nabla_w L = w - \sum \alpha_k \ell_k x_k \qquad \frac{\partial L}{\partial b} = \sum \alpha_k l_k = 0 \qquad \frac{\partial L}{\partial \xi_k} = C - \alpha_k - \beta_k = 0$$

Comme précédemment, on calcule la fonction duale H, et on obtient l'expression du problème dual : (à faire par vous-mêmes)

$$\begin{cases} \underset{\alpha \geq 0}{Max}\left(H(\alpha)\right) \\ \sum_{1}^{p} \alpha_k l_k = 0, \quad 0 \leq \alpha_k \leq C \end{cases}$$

Et donc la seule modification porte sur la contrainte : cette fois a doit être dans une « boite ». La méthode de projection est la même (adaptez-la)

Pour le calcul de b : attention

- Comme précédemment, identifier les points x_i pour lesquels α_i est non nul
- Pour ces points, la contrainte correspondante est active donc $1 - \xi_i - l_i\left(\langle w, \hat{x}_i \rangle + b\right) = 0$
- On ne peut pas calculer b, à cause du terme ξ_i
- On remarque que, comme $\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0$, si $\alpha_i \neq C \ \ \beta_i \neq 0$ et donc $\xi_i = 0$, ce qui permet alors de calculer b

2) Aide sur l'algorithme d'UZAWA à implémenter :

On écrira sur papier un algorithme de type Uzawa : maximisation de la fonction duale H(α) (minimisation de – H) par une méthode de gradient (traditionnellement : gradient à pas constant) de la fonction duale, avec contraintes inégalité/égalité linéaires, mais cependant plus complexes que les contraintes vues sur le cas standard (on a ici en plus une contrainte égalité : $\sum_{i=1,p} l_i \alpha_i = 0$)

Méthode de gradient avec projection : on fera un pas de gradient à pas constant dans R^p pour passer de $\alpha^k = (\alpha_i)^k$ à α^{k+1}, **suivi** d'une projection de l'itéré α^{k+1} sur l'espace des contraintes.

Etude empirique de la projection sur l'espace des contraintes que l'on notera C.

- Décrire géométriquement C
- Faire un dessin en dimension de 2 de C, faire marcher graphiquement (avec un crayon…) l'algorithme d'Uzawa en dimension 2. En même temps, calculer la solution analytique en dimension 2.
- On propose d'obtenir la projection en deux étapes : projection sur l'hyperplan ($\sum_{i=1,p} l_i \alpha_i = 0$) ***suivie*** de la projection sur le « quadrant » positif : $\alpha_i \geq 0$. Faites-le à la main sur votre dessin en dimension 2. Faites le contraire (projeter d'abord sur le quadrant positif). Conclusion ?
- En dimension n, cette « méthode de projection » n'est pas exacte. C'est une approximation…. Peut-on imaginer un algorithme simple pour la projection exacte (question ouverte….) ?

- Essayer de faire la même chose en 3D. L'hyperplan n'est pas positionné n'importe comment, le vecteur normal $l = (l_i)_{i=1\ldots p}$ (labels) étant un vecteur constitué de 1 et de – 1
- On utilisera cette décomposition dans le code, et on la testera empiriquement.
- Dans le cas convexe (contraintes et fonction objectif), la convergence d'une méthode gradient + projection (ie trouver le point le plus proche ….) sur les contraintes assure-t-elle l'obtention de la solution (ie : converge-t-on vers la solution ?)
- Ecrire la formule de projection sur l'hyperplan $\sum_{i=1,p} l_i \alpha_i = 0$

Une fois le problème résolu dans l'espace dual et les multiplicateurs de $(\alpha_i)_{i=1,p}$, quelle est l'expression de la fonction de décision ?

Aide pour l'obtention de b:

Les points supports sont ceux pour lesquelles la contrainte de marge est active , c'est-à-dire que le multiplicateur de Lagrange correspondant est non nul et que :

$1 - l_k \left(\langle w, \hat{x}_k \rangle_H + b \right) = 0$

De là on tire une valeur de b pour chaque point support, qui théoriquement est la même, aux erreurs numériques et d'approximation près. On prendra la moyenne de ces valeurs de b pour minimiser l'erreur.

3) Construction d'un logiciel de SVM (materiel élèves sur Campus)

On vous fournit pour cela quelques fonctions Matlab et un programme principal svm.mat

- Fichiers *data.m* : ils contiennent les données d'apprentissage, sous la forme de 2 tableaux X et lab .
    - *X* de dimension (n,p) contient les coordonnées des points d'apprentissage (on reste dans R^2) Chaque colonne de X est donc un point d'apprentissage dans R^n.
    - *lab* de dimension (p,1) contient les « labels » l_k: -1 pour les points du premier groupe, +1 pour les points du second groupe.
    - *Pour charger les données : load data1 X lab*
- Petit programme *makedata* qui vous permet de créer vous-mêmes, à la souris, des fichiers data
- Fonction *kernel* qui renvoie K(u,v) pour un couple u,v de vecteurs (colonnes) de R^n
- Fonction *prodw* qui calcule, à l'aide du noyau et des données (X,l), le produit scalaire de w avec un vecteur u.

Ces deux tableaux de données d'apprentissage sont stockés dans les fichiers data1, data2, data3…. qui serviront à ce que tout le monde travaille sur les mêmes cas, mais vous pouvez en fabriquer d'autres si besoin. Pour cela, on vous fournit le petit programme *makedata*

Dans makedata, les fichiers data1 … sont créés par la commande : *save 'datanew' X lab* que vous renommerez à votre guise, et seront donc éventuellement chargés par la commande (attention, le nom des variables doit être celui-ci)
load 'data1' X lab

Structure du code :

Elle peut être simple pour commencer :

- Déjà écrit : lecture et dessin des données (dessin dans le cas bidimensionnel, on se limitera à priori à ce cas)

Cœur du programme :

- assemblage de la matrice de la forme quadratique du problème dual
- choix d'un point de départ, du pas , d'un critère d'arrêt…
- boucle de l'algorithme d'Uzawa jusqu'à ce que le critère d'arrêt soit satisfait
- On obtient alors le vecteur α , il ne manque donc plus que la constante b pour déterminer complètement la fonction de décision
- Détermination de b à partir des points supports.

- Déjà programmé pour vous : le dessin de la ligne de niveau 0 de la fonction de décision, et des lignes de niveau 1 et -1 marquant les marges (dans le cas 2D)

Aide : pour les jeux de données data0 et data1, un « pas » de 10^{-3} (constante de la méthode du gradient à pas constant) est convenable….

On vous fournit la partie « dessin du résultat » : il suffit d'utiliser la fonction contour de Matlab sur la fonction de décision que l'on aura calculée sur un quadrillage, et de demander le traçage des isovaleurs -1, 0 et 1 .

Tests : on prendra bien soin de tester le code : tester pour commencer sur un ensemble de deux points, comparer avec la solution analytique, étudier la précision.

Puis on devra faire une étude numérique : influence des paramètres numériques : pas du gradient, point de départ, critère d'arrêt…. Sur l'efficacité de l'algorithme (comment juger l'efficacité ?)

Séparation non linéaire

Sur un plan pratique, il suffit « d'encapsuler » le produit scalaire dans une fonction (déjà fait d'ailleurs), que l'on pourra modifier pour le remplacer par un noyau. Exemple : le noyau gaussien . Attention : ce produit scalaire intervient dans la matrice A **et** dans la fonction de décision.

$$\left\langle \Phi(x), \Phi(y) \right\rangle_E = k(x, y) = \exp\left(-\frac{\|x - y\|^2}{\sigma^2} \right)$$

Les jeux de données 3 et 4 fournis contiennent en dimension 2 des données non linéairement séparables.

Présence d'*outliers* , marges « souples »

Le fichier data4 contient un exemple avec 3 *outliers*.
Modifier votre code en conséquence.
Le tester d'abord sur un fichier avec des données séparables linéairement (data1) et on verra varier la « souplesse » de la marge. Qu'observez-vous ?

Le tester à présent sur le fichier data4. Que se passerait-t-il avec ces données si on n'introduisait pas la technique « marge souple » ?

Pour finir et dans tous les cas, on permettra au logiciel construit de recevoir un nouveau point et de le classer automatiquement dans l'une des deux classes (fonction de décision)

Annexe : Partie 2 pour info : les SVR

Document : HAL : Méthodes SVM pour l'identification , Fabien Lauer, Gérard Bloch

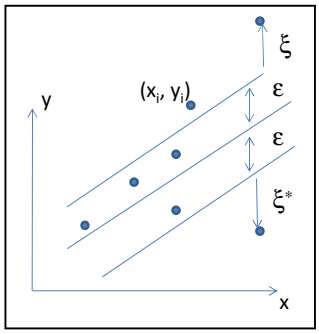
Le principe et la réalisation de cette partie est très proche de la précédente, (surtout des SVM avec marge floue). Il faudra cependant ré-écrire soigneusement les formulations primales et duales du problème d'optimisation, qui sera également résolu numériquement par la méthode d'UZAWA.

Le modèle est très bien expliqué dans le document fourni, dont vous devez lire la section 2.

Pour résumer

L'objectif est de développer un outil de régression, ou autrement dit de trouver une fonction de R^n dans R, y(x), approchant « au mieux » un ensemble de p couples (x_i, y_i) , x_i dans R^n , y_i dans R (échantillon d'apprentissage). Par rapport aux SVM : les y_i remplacent les labels l_i.

On commence par la régression linéaire, comme pour les SVM linéaires :



On cherche l'hyperplan (la droite sur le dessin ci-contre) qui approche au mieux les p points, au sens où on voudrait que ces p points soient à l'intérieur d'une marge de taille 2 ε (voir dessin), avec ε fixé (petit… c'est un paramètre de la méthode)

Pour permettre à certains points d'être en dehors de la marge, on introduit (Comme pour les marges souples) des variables d'écart $\xi_i > 0$, mais cette fois
2 pour chaque point, pour lui permettre d'être d'un côté ou de l'autre de la marge.

a) On veut bien sûr que ces variables prennent les valeurs les plus petites possibles.

b) On cherche le plan $y = \langle w, x \rangle + b$ le plus « horizontal » possible, c'est-à-dire celui dont le gradient *w est de norme la plus petite possible*. (On peut se poser la question de la raison de cette exigence. Elle est liée à la minimisation de la complexité du modèle)

Pour régler le compromis entre les exigences a et b, on introduit un coefficient réglable C et on veut donc résoudre le problème d'optimisation primal suivant :

$$Min \frac{1}{2}\|w\|^2 + C\sum\left(\xi_i + \xi_i^*\right) \quad \forall k = 1 \ldots p,$$

$$y_k - \left(\langle w, x_k \rangle + b\right) \le \varepsilon + \xi_k$$

$$y_k - \left(\langle w, x_k \rangle + b\right) \ge -\varepsilon - \xi^*_k$$

$$\xi_k \ge 0, \xi^*_k \ge 0$$

Une fois résolu, on a donc le « modèle » (ou « fonction apprise ») $y(x) = \langle w, x \rangle + b$
Le produit scalaire étant dans H si on est dans le cas non linéaire avec le *kernel trick*

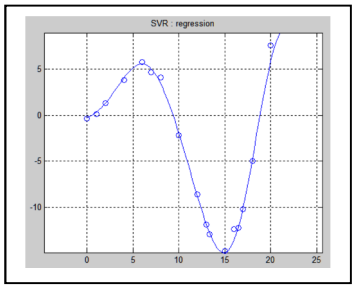
A partir de là, avec l'aide du document fourni (Méthodes SVM pour l'identification) et forts de votre expérience sur les SVM, écrivez le Lagrangien, le problème dual, programmez la méthode d'UZAWA (modifiez un peu le programme écrit précédemment pour les SVM).

Une difficulté signalée peut être pour la détermination de b. Un peu comme précédemment mais pas tout à fait. Réfléchissez !

De la même façon que précédemment, mettez en œuvre le *kernel trick* pour pouvoir traiter le cas où la fonction cherchée n'est pas un plan mais une nappe courbe (pas de difficulté particulière).

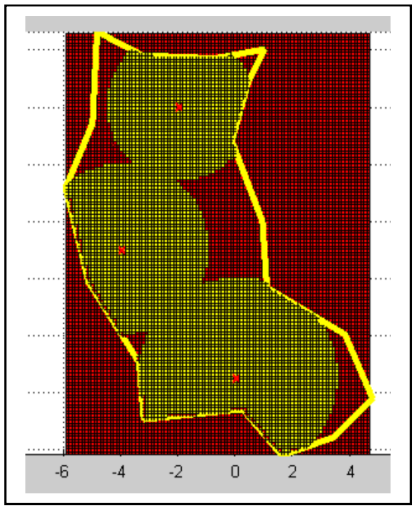
La fonction apprise sera alors : $y(x) = \langle w, x \rangle_H + b$

On prendra soin de stocker dans un fichier tout le matériel nécessaire à la reconstruction de cette fonction, de façon à pouvoir y faire appel sans repasser par une résolution SVR (cette fonction est simplement une combinaison linéaire pondérée de fonctions de base construites à partir du noyau, on stocke donc les poids, le noyau, les points x_i,yi, b ….



On pourra mettre le programme au point au début sur des fonctions de R dans R (il faudra probablement modifier les dessins proposés dans le code fourni qui fonctionnent pour des fonctions de R^2 dans R)

Exemple de réalisation en dimension 1 pour x :



Les antennes dans la Loire

On cherche la position optimale de p antennes emmenant sur des rayons donnés, pour couvrir le plus de surface possible sur le département de la Loire.
Un logiciel permet de calculer la surface couverte en fonction de des position C_i i=1 …p des p antennes (centre des cercles) , les rayons étant fixés.

On commence par une seule antenne mobile, et une antenne dont la position est fixée. Ce cas correspond au fichier d'apprentissage *antennes_2d_train.mat*
(*Attention*, le tableau que l'on a nommé X jusqu'à présent s'appelle C dans le fichier des points d'apprentissage et est transposé par rapport à X.)

A partir de ce fichier, et par la méthode des SVR (noyau gaussien), on générera une surface de remplacement sous la forme d'une fonction f (x) , x étant le vecteur des centres, de dimension 2xnombre d'antennes (au début donc de dimension 2)

Pour générer f(x), il suffit de connaître les points d'apprentissage x_i, les coefficients α_i et la constante b. On pourra stoker ce matériel dans un fichier modele.mat par exemple pour bâtir la fonction f(x) indépendamment du code de SVR (après coup).

Puis on utilisera cette fonction dans un programme d'optimisation locale. On analysera les résultats à partir de plusieurs points de départ.

Enfin, on fournit un fichier antennes_2d_test.mat qui contient des positions de centres autres que celles des points d'apprentissage. On vous demande d'évaluer votre fonction apprise f sur cet échantillon de points, et de conserver le résultat dans un fichier, pour comparaison ultérieure avec le code de calcul de surfaces de couverture originel (servira pour votre évaluation sur ce TP)

Un peu d'aide tout de même pour les SVR :

Comme vous le verrez et devez le comprendre (refaites les calculs), la fonction duale s'écrit :

$$H(\alpha,\alpha^*) = -\frac{1}{2}\sum_{i,j=1..p}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left\langle x_i,x_j\right\rangle - \varepsilon\sum_{k=1..p}\left(\alpha_k+\alpha_k^*\right)+\sum_{k=1..p} y_k\left(\alpha_k-\alpha_k^*\right)$$

Le vecteur des variables duales, inconnue du problème, alpha est deux fois plus long que dans le cas des SVM (vecteur colonne)

On note $\alpha = (\alpha_1,\alpha_2,....\alpha^*_1,\alpha^*_2....)'$

Si on note B la matrice pxp $B_{ij} = \left\langle x_i,x_j\right\rangle$, quelle est la matrice A 2px2p

$$\alpha' A\alpha = \sum_{i,j=1..p}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left\langle x_i,x_j\right\rangle$$

On écrira les deux termes linéaires sous la forme $\left\langle u_1,\alpha\right\rangle$ et $\left\langle u_2,\alpha\right\rangle$, u& et u 2 dans R^{2p}

Calcul du terme b :

Comme pour les SVM, il faut se servir des points supports…. *Mais pas tous*

Comme précédemment, on identifie les points support , pour lesquels le multiplicateur de Lagrange a ou a* es nul et donc

$$y_k - \left(\left\langle w,x_k\right\rangle + b\right) - \varepsilon - \xi_k = 0$$

ou

$$-y_k + \left(\left\langle w,x_k\right\rangle + b\right) - \varepsilon - \xi^*_k = 0$$

Cela permet d'obtenir b qui si on connait les variables x ou x* …. Quels sont les points supports pour lesquels on connait ces variables …. Ou encore : quand ces variables sont-elles nulles ?

Ne restez pas bloqués en séance, posez des questions vous serez aidés.