TP Optimisation classique

Ce TP a pour objectif de mettre en œuvre les méthodes d’optimisation contrainte, et plus particulièrement les méthodes numériques duales, pour construire un outil de classification automatique : « Machine a Vecteur Support ». Tous les éléments de codes sont donnés dans le langage Matlab.

# Analyse linéaire simple

**Montrer que la distance d’un point à l’hyperplan vaut ou encore si on oriente correctement w.**

Soit P un point dont les coordonnées sont données par. On suppose qu’il n’appartient pas à l’hyperplanH d’équation . Soit Q un point ∈H, de coordonnées *q=(q1, …, qn)*,tel que soit orthogonal à l’hyperplan (projection orthogonale sur l’hyperplan). On cherche donc la distance |, qui correspond à la distance du point au plan. On a :

| = =

Or H d’où

| = = = distance de P à l’hyperplan

Si on oriente vers l’espace des cellules saines (i.e. ), alors la distance à l’hyperplan peut s’écrire

**Montrer que l’ensemble des points vérifiant sont situés hors d’une marge de demi-largeur , que l’on voudra donc maximiser.**

Soit la demi-marge, le point est situé hors de cette marge si sa distance à l’hyperplan est plus grande que sa valeur. D’où

Lorsque les points sont linéairement séparables, il existe généralement une infinité de frontières, et donc de fonctions de décision. La recherche d’optimisation de l’hyperplan H, i.e. telle que la marge soit maximisée, permet de déboucher sur la recherche d’un unique couple (w,b). Le problème est donc mieux posé lorsque que cette contrainte est prise en compte.

**Exprimer le problème comme un problème de minimisation quadratique avec contraintes linéaires**

On cherche à maximiser la marge à l’hyperplan, donc.   
On peut ainsi poser la recherche de minimisation suivante, avec ses contraintes :

* Fonction coût :
* Contraintes :

En introduisant le vecteur des multiplicateurs de Kuhn et Tucker , on en déduit la forme du Lagrangien suivante :

**(1)**

Le calcul des gradients du Lagrangien en w et b nous permettent d’obtenir les conditions d’optimalité de Kuhn et Tucker du programme quadratique ci-dessus.

On a :

**(3)**

En posant la fonction duale, et en considérant les égalités obtenues grâce aux conditions d’optimalité, on trouve :

En utilisant **(2)**

En utilisant **(3)**

En posant la matrice , on obtient

La fonction duale H est donc de forme quadratique, et strictement concave. On la rend convexe en changeant son signe (i.e. multiplication par le facteur -1). La matrice A étant symétrique, la forme quadratique est définie positive. Cela assure la convergence du modèle vers l’unique solution.

En appliquant ces résultats à la recherche de l’optimum du SVM, et en changeant le signe de la fonction H, on obtient le problème dual suivant :

# Rédaction du code :

Initialisation des données :

alph0=0.5\*ones(na,1); % point de départ (0, ou autre): réglable

pasgrad=5e-3; % pas du gradient : parametre réglable

u=ones(na,1); % vecteur de 1

crit\_arret=1; % initialisation critere d'arret

npas=0; % comptage du nombre de pas

npas\_max=100000; % garde-fou convergence (si ça ne converge pas...)

epsi=1e-5; % seuil convergence

Création de la matrice A (notée M dans le code) :

%Conception de la matrice M

A=ones(na,1);

M=A\*A'; %On conçoit ici la matrice de dim (na\*na, remplissage par des 1), que l'on va maintenant remplir

fori=1:na

for j=1:na

M(i,j)=M(i,j)\*lab(i)\*lab(j)\*X(:,i)'\*X(:,j);

end

end

%On a bien H(alpha) = -1/2\*alphaT.\*M.\*alpha - AT.\*alpha

Méthode du gradient pour obtenir le minimum :

while and(crit\_arret>epsi,npas<npas\_max)

npas=npas+1;

%H(npas)=sum(alph0)-1/2\*sum(sum(M));

gradH=-(M\*alph0-u); %Calcul du gradient

alph=alph0+pasgrad\*gradH;

alph=alph-alph'\*lab/(lab'\*lab)\*lab%Contrainte sur la somme des alph\*lab

alph = max(alph, zeros(na,1)); %Projection sur R4+

%Test de convergence

Conv=norm(alph-alph0)

if Conv<epsi

crit\_arret = 0

end

alph0=alph;

%Si critère de convergence atteint, on modifie le critère d'arret pour

%sortir de la boucle

ifConv<epsi

crit\_arret=epsi/2 ;

end

end

Calcul de w et de b grâce à l’optimum trouvé. Concrètement, on obtient w grâce à la formule (2), b est obtenu légèrement différemment. On part du constat que tous les points supports (i.e. interceptés par la marge), vérifie l’égalité . On choisit donc de la valeur trouvée pour b sur l’ensemble des points.

w=zeros(2,1)

fori = 1:na

w=w+ alph0(i)\*lab(i)\*X(:,i)

end

Points\_support=find(alph>epsi)'

%Si le point constitue une contrainte active, alors on a b = 1 - lk\*w'\*xk

b=0;

fori=Points\_support

b=b+1/lab(i)-w'\*X(:,i);

end

b=b/length(Points\_support);

Obtention de la fonction f en appliquant sa définition:

xp=xmin:0.2:xmax; % cr¨¦ation d'une grille

yp=ymin:0.2:ymax;

npx=length(xp);

npy=length(yp);

fori=1:npx

for j=1:npy

% calcul de <w,x> + b sur une grille

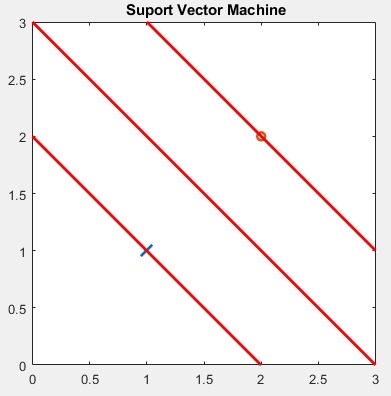
x=[xp(i);yp(j)];

V(i,j)= b + w'\*x;

end

end

On teste le code sur un échantillon de 2 points, dont un se trouve de chaque côté de la marge. Cela nous permettra de vérifier son fonctionnement et d’évaluer sa précision. On obtient :



Les points ont pour coordonnées respectivement (2,2) et (1,1). Dans ce cas trivial, la solution est la médiatrice du segment qui relie les deux points. Les marges, également représentées sur le graphique, passent bien par les points et sont perpendiculaire à la fonction de décision, qui est la droite du milieu.

En théorie, pour la fonction de décision   
f : x → , on devrait obtenir ici b = -3, et w=(1,1).

Le code renvoie b=-2.9958 et w=(0.9986, 0.9986)

Les résultats sont donc très proches de la théorie, et valident les hypothèses de critère de convergence retenues dans le code

Figure 1- Test avec 2 points

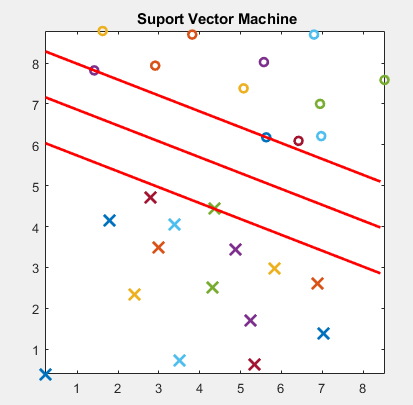
On utilise désormais un panel de point plus important, par exemple data1. On obtient le résultat suivant :

Figure 2 - SVM linéaires data1

Etude de l’influence des paramètres :

On affiche le nombre de pas effectués (dont le temps d’exécution est proportionnel) avant d’obtenir le point optimal en fonction du pas choisi. On observe une convexe, dont le minimum est atteint aux alentours de 5.10-3. Un pas trop petit, ou trop grand engendre une instabilité du programme, qui ne converge plus.

Une étude similaire, mais cette fois-ci en faisant fluctuer le point de départ, montre que ce dernier n’a pratiquement aucun impact sur le nombre de pas effectués.

Etude de l’influence du critère de convergence (i.e. de la différence entre ) :

Sans surprise, on constate que plus le critère est petit (donc contraignant), plus le nombre d’exécution est important. Il croit de manière significative dès que l’on descend en dessous de 1E-6. Le graphique ci-dessous nous en donne un aperçu.

Il faut donc choisir judicieusement l’ensemble de ces critères pour maximiser l’efficacité du programme selon les attentes de résultats.

# Etude des SVM dans le cas de la présence d’outliers.

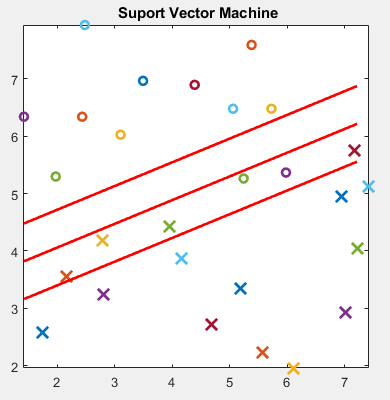
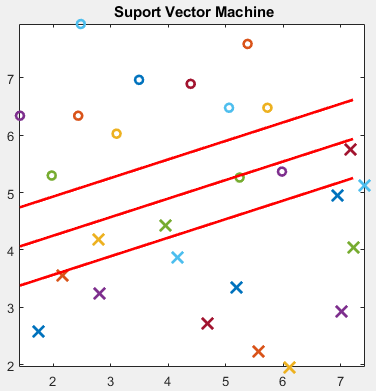
Avec le fichier data 3, on obtient les résultats suivants

Figure 3 - Marge pour C = 1

Figure 4 - Marge pour C = 0.5

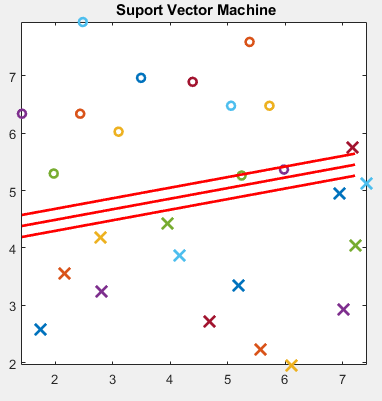
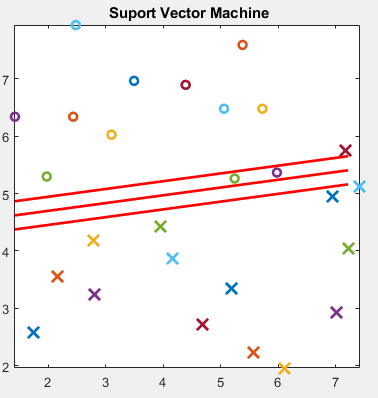


Figure 5 - Marge pour C = 10

Figure 6 - Marge pour C = 20

L’exécution du programme sans l’ajout de marges « souples » ne converge pas. On ajoute donc ce critère uniquement grâce à cette ligne dans le code :

MatC=C\*ones(na,1);

alph = min(alph, MatC);

On constate donc que plus C est grand, moins les marges sont larges. A partir d’une certaine valeur de C, elles ne varient plus.

Etape 5 :

Séparation non linéaire :

D’aprés des démonstration qu’on a déjà fait, on peut facilement obtenir la cas de séparation non linéaire.

On va remplacer avec et on peux obtenir le problème dual est :

Et on utilise le noyaux gaussien,

Donc , après la programmation, on a testé avec data3.mat et des de valeurs différentes, et obtenu les figures :

Quand on défine :

Ça peut juste bien séparer les 2 types de points.

Quand on défine :



Et quand on define :



Avec l’augmentation de , la séparation est de moins en moins stricte.

Et on utilise des point aléatoires data6.mat obtenus par makedata.m , et test quand :



Il peut aussi strictement séparer des points. Donc le truc peut être très bien marché.

Conclusion :

D’après nos études, on peux voir :

Quand on peut faire une séparation linéaire, si on utilise outliners, on peut conclure que plus C est grand, moins les marges sont larges. A partir d’une certaine valeur de C, elles ne varient plus.

Quand les points ne peuvent pas être séparés par séparation linéaire. On va utiliser noyaux pour établir une séparation non linéaire. On peut obtenir par SVM non linéaire, quand on augmente , la séparation est de moins en moins stricte, quelques points ne peuvent pas être séparé par des plans.