Data Mining 2
TP 2.4 Multi-class SVM

Thomas Robert

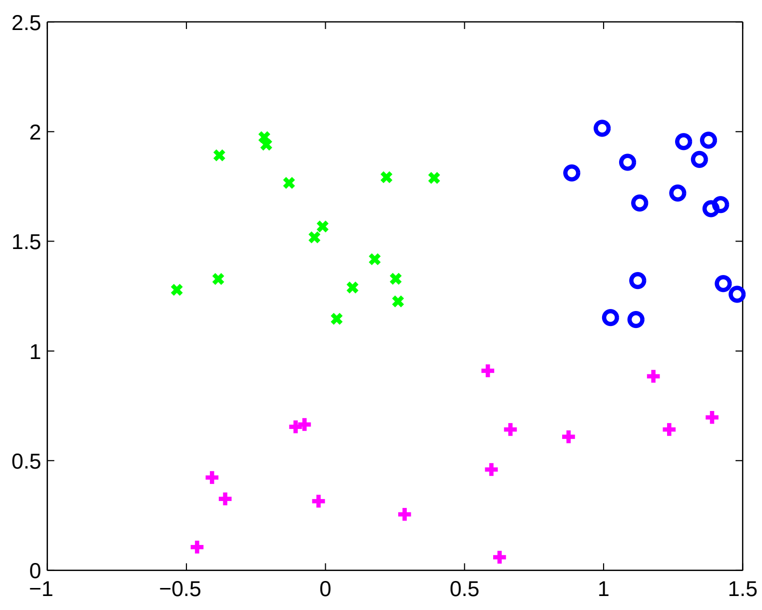
Introduction

L'objectif de ce TP est de réaliser un SVM 3 classes par diverses méthodes : linéaire 1 vs all, linéaire all together, avec kernel.

Dataset

On génère un jeu de données

```
ni = 15;
of = 1;
X1 = rand(ni,2) ;
X1(: ,1) = 2*X1(: ,1) -.5;
X2 = rand(ni,2) + of*ones(ni,1) *[.55 1.05];
X3 = rand(ni,2) + of*ones(ni,1) *[ -.55 1.05];
Xi = [X1;X2;X3];
[n,p] = size(Xi) ;
yi = [[ones(ni,1) ; -ones(ni,1) ; -ones(ni,1) ] , [ -ones(ni,1) ; ones(ni,1) ; ...
    -ones(ni,1) ] , [ -ones(ni,1) ; -ones(ni,1) ; ones(ni,1) ]];
yii = [ones(ni,1) ; 2*ones(ni,1) ; 3*ones(ni,1) ];
nt = 1000;
X1t = rand(nt,2) ;
X1t(: ,1) = 2*X1t(: ,1) -.5;
X2t = rand(nt,2) + of*ones(nt,1) *[.55 1.05];
X3t = rand(nt,2) + of*ones(nt,1) *[ -.55 1.05];
Xt = [X1t;X2t;X3t];
yt = [ones(nt,1) ; 2*ones(nt,1) ; 3*ones(nt,1) ];
plot(X1(: ,1) ,X1(: ,2) , '+m' , 'LineWidth' ,2) ; hold on
plot(X2(: ,1) ,X2(: ,2) , 'ob' , 'LineWidth' ,2) ;
plot(X3(: ,1) ,X3(: ,2) , 'xg' , 'LineWidth' ,2) ;
```

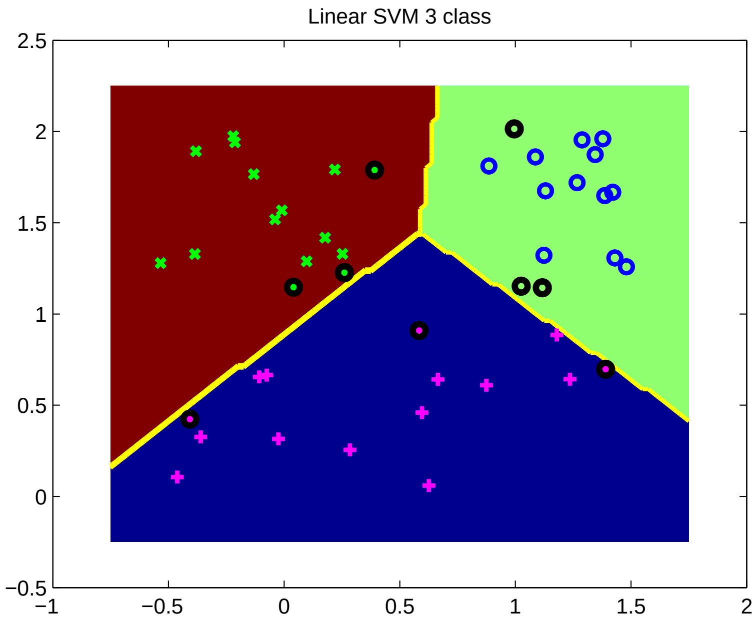


Linéaire 1 vs all

On test d'abord une approche linéaire 1 vs all. On obtient un taux d'erreur de 10%.

```
kernel= 'poly'; d=1;
C = 1000000000;
lambda = 1e-8;
[xsup1 ,w1,b1,ind_sup1 ,a1] = svmclass(Xi,yi(: ,1) ,C,lambda ,kernel ,d,0) ;
[xsup2 ,w2,w02,ind_sup2 ,a2] = svmclass(Xi,yi(: ,2) ,C,lambda ,kernel ,d,0) ;
[xsup3 ,w3,w03,ind_sup3 ,a3] = svmclass(Xi,yi(: ,3) ,C,lambda ,kernel ,d,0) ;

% support vector
vsup = [ind_sup1; ind_sup2; ind_sup3];

% test
ypred1 = svmval(Xt,xsup1 ,w1,w01,kernel ,d) ;
ypred2 = svmval(Xt,xsup2 ,w2,w02,kernel ,d) ;
ypred3 = svmval(Xt,xsup3 ,w3,w03,kernel ,d) ;
[v yc] = max([ypred1 , ypred2 , ypred3]') ;

% error rate
nbbienclasse = length(find(yt ==yc') );
freq_err = 1 - nbbienclasse/(3*nt)

% plot
[xtest1 xtest2] = meshgrid([ -0.75:.025:1.75] ,[-.25:0.025:2.25]) ;
[nnl nnc] = size(xtest1);
Xtest = [reshape(xtest1 ,nnl*nnc ,1) reshape(xtest2 ,nnl*nnc ,1) ];
ypred1 = svmval(Xtest ,xsup1 ,w1,w01,kernel ,d) ;
ypred2 = svmval(Xtest ,xsup2 ,w2,w02,kernel ,d) ;
ypred3 = svmval(Xtest ,xsup3 ,w3,w03,kernel ,d) ;
[v yc] = max([ypred1 , ypred2 , ypred3]') ;
ypred1 = reshape(ypred1 ,nnl,nnc) ;
ypred2 = reshape(ypred2 ,nnl,nnc) ;
ypred3 = reshape(ypred3 ,nnl,nnc) ;
yc = reshape(yc,nnl,nnc) ;
contourf(xtest1 ,xtest2 ,yc,50) ; shading flat; hold on
plot(X1(: ,1) ,X1(: ,2) , '+m' , 'LineWidth' ,2) ;
plot(X2(: ,1) ,X2(: ,2) , 'ob' , 'LineWidth' ,2) ;
plot(X3(: ,1) ,X3(: ,2) , 'xg' , 'LineWidth' ,2) ;
h3=plot(Xi(vsup ,1) ,Xi(vsup ,2) ,'ok' , 'LineWidth' ,2) ;
[cc, hh]=contour(xtest1 ,xtest2 ,yc,[1.5 1.5] , 'y-' , 'LineWidth' ,2) ;
[cc, hh]=contour(xtest1 ,xtest2 ,yc,[2.5 2.5] , 'y-' , 'LineWidth' ,2) ;
plot(X1(: ,1) ,X1(: ,2) , '+m' , 'LineWidth' ,2) ; hold on
plot(X2(: ,1) ,X2(: ,2) , 'ob' , 'LineWidth' ,2) ;
plot(X3(: ,1) ,X3(: ,2) , 'xg' , 'LineWidth' ,2) ;
h3=plot(Xi(vsup ,1) ,Xi(vsup ,2) ,'ok' , 'LineWidth' ,3) ;
title('Linear SVM 3 class');
```

```
freq_err =

    0.1030
```


Linear SVM 3 class
2.5
2
1.5
1
0.5
0
−0.5
−1
−0.5
0
0.5
1
1.5
2

All together

On teste ensuite la méthode all together qui consiste à résoudre un seul problème de minimisation pour trouver plusieurs SVM.

On notera que ce problème contient énormément de contraintes.

On obtient un taux d'erreur de 7%. Cependant, lors des tests, on trouve généralement un moins bon résultat avec cette méthode qu'avec l'approche 1 vs all.

```
cvx_begin
    variables w1(p) w2(p) w3(p) b1(1) b2(1) b3(1)
    dual variables lam12 lam13 lam21 lam23 lam31 lam32
    minimize( .5*(w1'*w1 + w2'*w2 + w3'*w3) )
    subject to
        lam12 : (X1*(w1-w2) + b1 - b2) >= 1;
        lam13 : (X1*(w1-w3) + b1 - b3) >= 1;
        lam21 : (X2*(w2-w1) + b2 - b1) >= 1;
        lam23 : (X2*(w2-w3) + b2 - b3) >= 1;
        lam31 : (X3*(w3-w1) + b3 - b1) >= 1;
        lam32 : (X3*(w3-w2) + b3 - b2) >= 1;
cvx_end

% error rate
ypred1 = Xt*w1 + b1;
ypred2 = Xt*w2 + b2;
ypred3 = Xt*w3 + b3;
[v yc] = max([ypred1 , ypred2 , ypred3]') ;
nbbienclasse = length(find(yt ==yc') );
freq_err = 1 - nbbienclasse/(3*nt)
```

```
freq_err =

    0.0720
```

All together matrix form

On écrit cette fois le problème "all together" de manière matricielle en primal puis en dual. Cela permet de résoudre ce problème avec monqp.

```
% all together primal

Z = zeros(ni,p) ;
X = [X1 -X1 Z;
     X1 Z -X1;
     -X2 X2 Z ;
     Z X2 -X2 ;
     -X3 Z X3;
     Z -X3 X3];
l = 10^ -12;
A = [1 1 -1 0 -1 0 ; -1 0 1 1 0 -1];
A = kron(A,ones(1 ,ni) ) ;

cvx_begin
    cvx_precision best
    %cvx_quiet(true)
    variables w(3*p) b(2)
    dual variables lam
    minimize( .5*(w'*w) )
    subject to
        lam : X*w + A'*b >= 1;
cvx_end


% all together dual

% compute G
K = Xi*Xi'; % kernel matrix
M = [1 -1 0; 1 0 -1 ; -1 1 0 ; 0 1 -1; -1 0 1; 0 -1 1];
```

```
MM = M*M';
MM = kron(MM,ones(ni) ) ;
Un23 = [1 0 0;1 0 0 ; 0 1 0 ; 0 1 0; 0 0 1 ; 0 0 1];
Un23 = kron(Un23,eye(ni) ) ;
G = MM.*(Un23*K*Un23') ;

% solve problem

l = 10^ -6;
I = eye(size(G) ) ;
G = G + l*I;
e = ones(2*n,1) ;
cvx_begin
    variables al(2*n)
    dual variables eq po
    minimize( .5*al'*G*al - e'*al )
    subject to
        eq : A*al == 0;
        po : al >= 0;
cvx_end

% monqp
[alpha , b, pos] = monqp(G,e,A' ,[0;0] ,inf,l,0) ;

% results
[al lam [lam12;lam13;lam21;lam23;lam31;lam32]]
```

```
ans =

    0.0000    0.0000    0.0000
    [...]
    0.0000    0.0000    0.0000
   16.3879   16.3882   16.3882
    0.0000    0.0000    0.0000
    [...]
    0.0000    0.0000    0.0000
    1.5504    1.5505    1.5505
    0.0000    0.0000    0.0000
    [...]
    0.0000    0.0000    0.0000
    1.1136    1.1136    1.1136
    0.0000    0.0000    0.0000
    [...]
    0.0000    0.0000    0.0000
   19.0519   19.0522   19.0523
    0.0000    0.0000    0.0000
    [...]
    0.0000    0.0000    0.0000
    5.1778    5.1778    5.1778
    0.0000    0.0000    0.00007
    [...]
    0.0000    0.0000    0.0000
    7.8418    7.8419    7.8419
    0.0000    0.0000    0.0000
    [...]
    0.0000    0.0000    0.0000
```

Kernelize multi-class SVM

On ajoute maintenant un kernel à notre calcul. Pour faire ce calcul, on crée 2 fonctions SVM3Class et SVM3val. Notons que ces fonctions sont perfectibles car ne fonctionnent que pour 3 classes de même taille.

Fonction SVM3class

```
function [Xsup, alpha, b] = SVM3Class(Xi, yi, C, kernel, kerneloption, options)

% note yi n'est pas utilisé pour simplifier les calculs. Normalement, il
% devrait influer sur la forme des matrices A, M et Un23.

[n, p] = size(Xi);
ni = n/3;

% matrice A
A = [1 1 -1 0 -1 0 ; -1 0 1 1 0 -1];
A = kron(A,ones(1, ni)) ;

% matrice M et MM
M = [1 -1 0; 1 0 -1 ; -1 1 0 ; 0 1 -1; -1 0 1; 0 -1 1];
MM = kron(M*M', ones(ni));

% matrice Un23
Un23 = [1 0 0;1 0 0 ; 0 1 0 ; 0 1 0; 0 0 1 ; 0 0 1];
Un23 = kron(Un23,eye(ni));

% calcul de G
K = svmkernel(Xi, kernel, kerneloption);
G = MM.*(Un23*K*Un23') ;

% QP
l = 10^ -5;
I = eye(size(G));
G = G + l*I; % kernel matrix
e = ones(2*n,1) ;
[al , ~, pos] = monqp(G, e, A', [0;0], C, l, 0);
n23 = 2*ni;

% calcul de b
yp = G(: ,pos) * al;
b1 = 1 - yp(pos(1)) ;
p2 = (find((pos > n23) & (pos <= 2*n23)));
b2 = 1 - yp(pos(p2(1)));
b3 = 1 - yp(pos(end));

b = [b1; b2; b3];

% calcul de alpha
alpha = zeros(2*n,1);
alpha(pos) = al;

Xsup = Xi;
```

Fonction SVM3val

```
function [y_pred] = SVM3Val(Xtest, Xsup, alpha, b, kernel, kerneloption)

[n, p] = size(Xsup);
n23 = 2/3*n;

% split b
b1 = b(1);
b2 = b(2);
b3 = b(3);

% split alpha
al12 = alpha(1:n/3) ;
al13 = alpha(n/3+1:n23) ;
al21 = alpha(n23+1:n23+n/3) ;
al23 = alpha(n23+n/3+1:2*n23) ;
al31 = alpha(2*n23+1:2*n23+n/3) ;
al32 = alpha(2*n23+n/3+1:end) ;

% compute kernel
```

```
K = svmkernel(Xtest, kernel, kerneloption, Xsup);

K1 = K(: ,1:n/3) ;
K2 = K(: ,n/3+1:n23) ;
K3 = K(: ,n23+1:end) ;

% predict
ypred1 = K1*al12 + K1*al13 - K2*al21 - K3*al31 + b1;
ypred2 = K2*al21 + K2*al23 - K1*al12 - K3*al32 + b2;
ypred3 = K3*al31 + K3*al32 - K1*al13 - K2*al23 + b3;

[~, yc] = max([ypred1 , ypred2 , ypred3]') ;

y_pred = yc;
```

Script de test

```
kernel = 'gaussian';
kerneloption = .25;

[Xsup, alpha, b] = SVM3Class(Xi, yi, C, kernel, kerneloption);


yc = SVM3Val(Xtest, Xsup, alpha, b, kernel, kerneloption);
yc = reshape(yc,nnl,nnc) ;

% affichage

figure;
colormap( 'autumn') ;
contourf(xtest1 ,xtest2 ,yc,50) ; shading flat; hold on
plot(X1(: ,1) ,X1(: ,2) , '+m' , 'LineWidth' ,2) ;
plot(X2(: ,1) ,X2(: ,2) , 'ob' , 'LineWidth' ,2) ;
plot(X3(: ,1) ,X3(: ,2) , 'xg' , 'LineWidth' ,2) ;

[cc, hh]=contour(xtest1 ,xtest2 ,yc,[1.5 1.5] , 'y-' , 'LineWidth' ,2) ;
[cc, hh]=contour(xtest1 ,xtest2 ,yc,[2.5 2.5] , 'y-' , 'LineWidth' ,2) ;
plot(X1(: ,1) ,X1(: ,2) , '+m' , 'LineWidth' ,2) ; hold on
plot(X2(: ,1) ,X2(: ,2) , 'ob' , 'LineWidth' ,2) ;
plot(X3(: ,1) ,X3(: ,2) , 'xg' , 'LineWidth' ,2) ;
title('SVM 3 class with gaussian kernel');
hold off
```