Data Mining
TP11 - Support Vector Machine

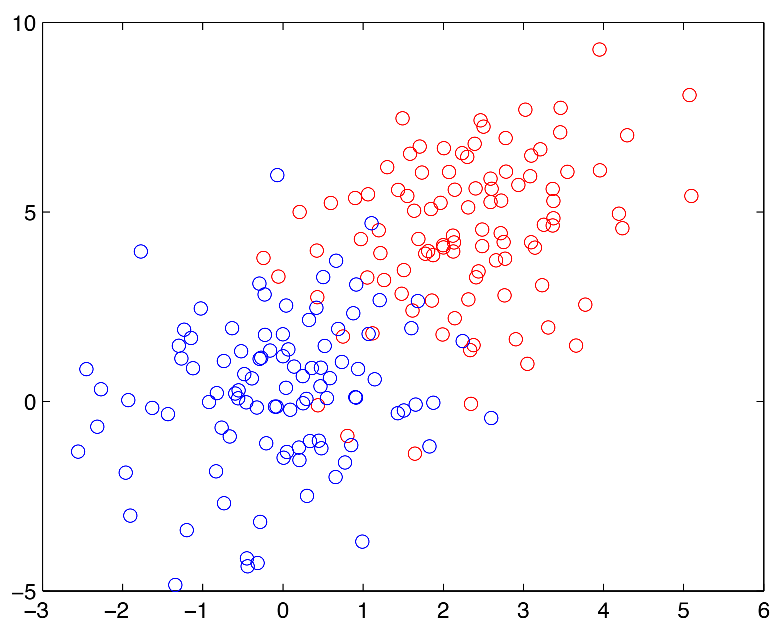
Rémi MUSSARD - Thomas ROBERT

Partie 2.1

Q1. Dataset

Construction du jeu de données

```
n = 100;

X1 = randn(n, 2);
X2 = randn(n, 2) + ones(n, 2)*2;
S = [1 0.5; 0.5 4];
X =[X1; X2]*S^(1/2);
y = [ones(n,1); -ones(n,1)];

figure;
plot(X(y==-1,1), X(y==-1,2),'or');
hold on
plot(X(y==1,1), X(y==1, 2),'ob');
```



Q2. Dual

Implémentation du problème de minimisation SVN dual sous CVX

```
C = 1000;
```

```

K = X*X';
Y = diag(y);
H = Y*K*Y;
q = ones(2*n, 1);

cvx_begin
    variable a(2*n)

    maximize(sum(a) - (1/2)*a'*H*a)

    subject to

        a>=0;
        C*q>=a;
        a'*y==0;

cvx_end

w = (a'*Y*X)'
```

```
w =

   -1.0586
   -0.5623
```

Q3. Frontiere

On peut obtenir b en trouvant la frontière de décision grâce aux points supports et en trouvant donc directement b.

On peut également implémenter directement le problème primal sous CVX comme ci-dessous qui donne directement b.

```
cvx_begin
    variable w(2)
    variable b(1)
    variable ksi(2*n)

    minimize(1/2 * w'*w + C * sum(ksi) )

    subject to

        y .* (X * w + b) >= 1 - ksi;
        ksi >= 0;

cvx_end

w
b
```

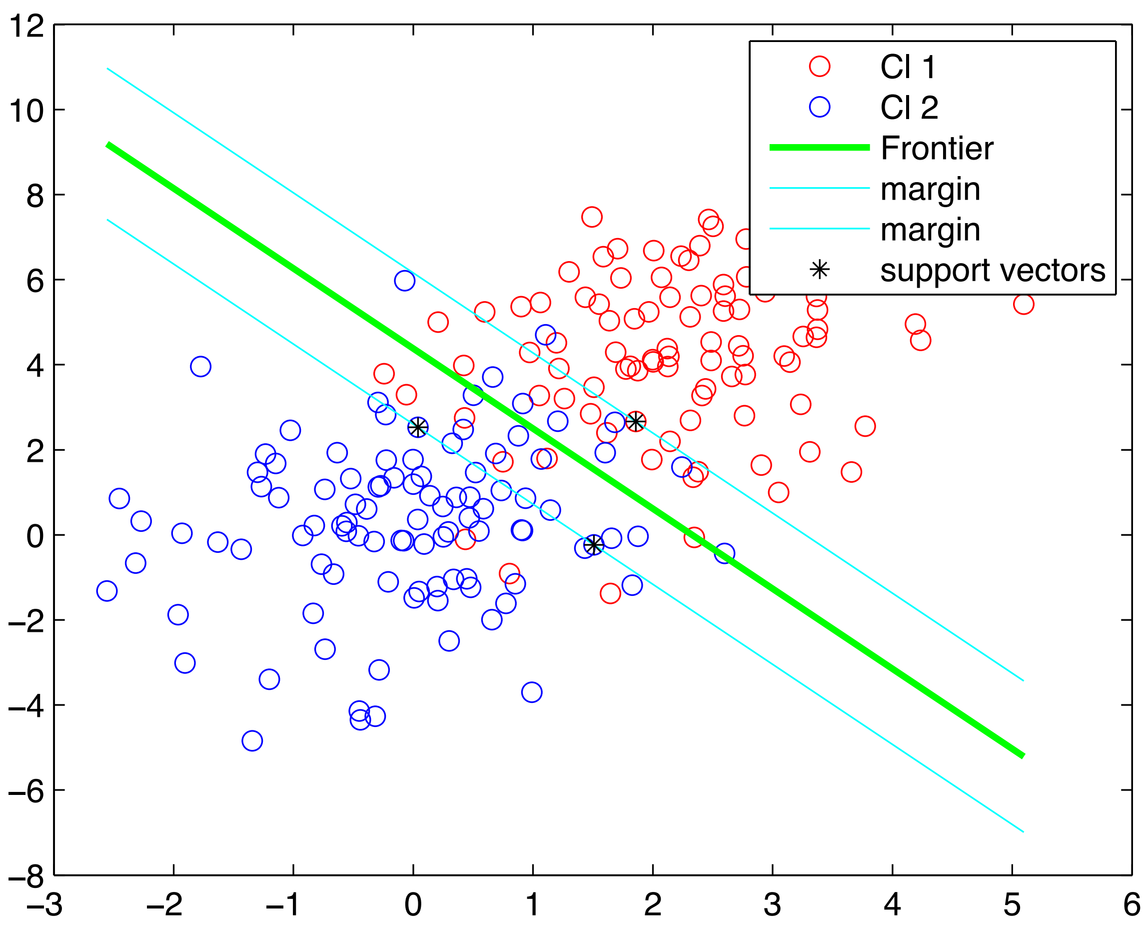
```
w =

   -1.0586
   -0.5623

b =

    2.4638
```

Q3. Isocontours

```
% Isocontour pour f(x) = 0

xFront = [min(X(:,1)) ; max(X(:,1))];
yFront0 = (-b-w(1)*xFront)/w(2);
plot(xFront, yFront0, '-g', 'LineWidth',2);

% Isocontour pour f(x) = 1

yFront1 = (1 -b-w(1)*xFront)/w(2);
plot(xFront, yFront1, '-c');

% Isocontour pour f(x) = - 1

yFrontm1 = (-1 -b-w(1)*xFront)/w(2);
plot(xFront, yFrontm1, '-c');
```

Q5. Points supports

```
support = single(y.*(X*w + b));
plot(X(support==1,1), X(support==1, 2),'*k');

legend('Cl 1', 'Cl 2', 'Frontier', 'margin', 'margin', 'support vectors')
```



Remarque

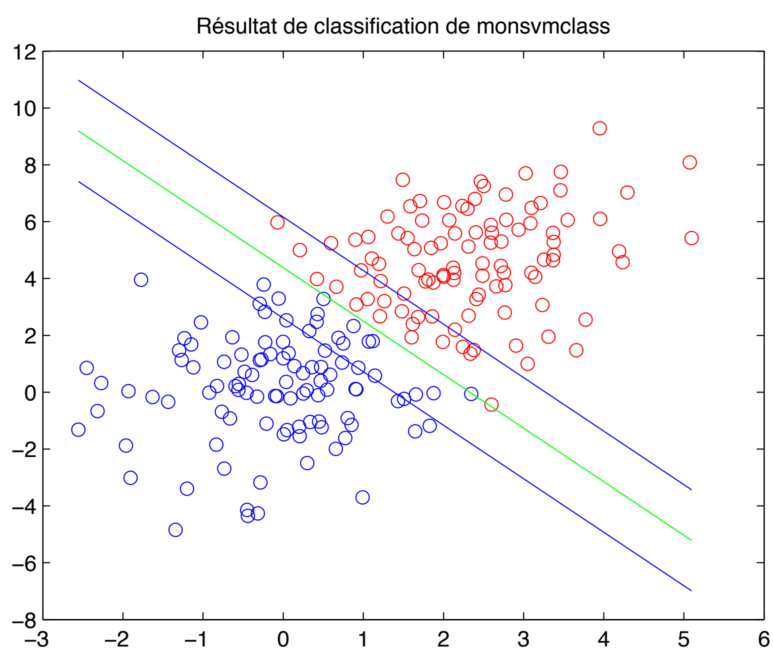
Avec cette méthode on se rend compte qu'il y a des erreurs de classification. Lorsqu'on a une donnée censé appartenir à une classe C1, avec la fonction monsvmval, on voit que cette donnée est affectée à la classe C2, car elle se trouve du mauvais côté de la frontière de décision.

Création des fonctions monsvmclass.m et monsvmval.m

Voir les fichiers monsvmclass.m et monsvmval.m

Vérification des fonctions sur les données initiales

```
[w, b, alpha] = monsvmclass(X, y, C);
w = w';
yReconst = monsvmval(X, w, b);

figure();
plot(X(yReconst==-1,1), X(yReconst==-1,2),'or');
hold on
plot(X(yReconst==1,1), X(yReconst==1, 2),'ob');
title('Résultat de classification de monsvmclass');

% Isocontour pour f(x) = 0

xFront = [min(X(:,1)) ; max(X(:,1))];
yFront0 = (-b-w(1)*xFront)/w(2);
plot(xFront, yFront0, '-g');

% Isocontour pour f(x) = 1

yFront1 = (1 -b-w(1)*xFront)/w(2);
plot(xFront, yFront1, '-b');

% Isocontour pour f(x) = - 1

yFrontm1 = (-1 -b-w(1)*xFront)/w(2);
plot(xFront, yFrontm1, '-b');
```

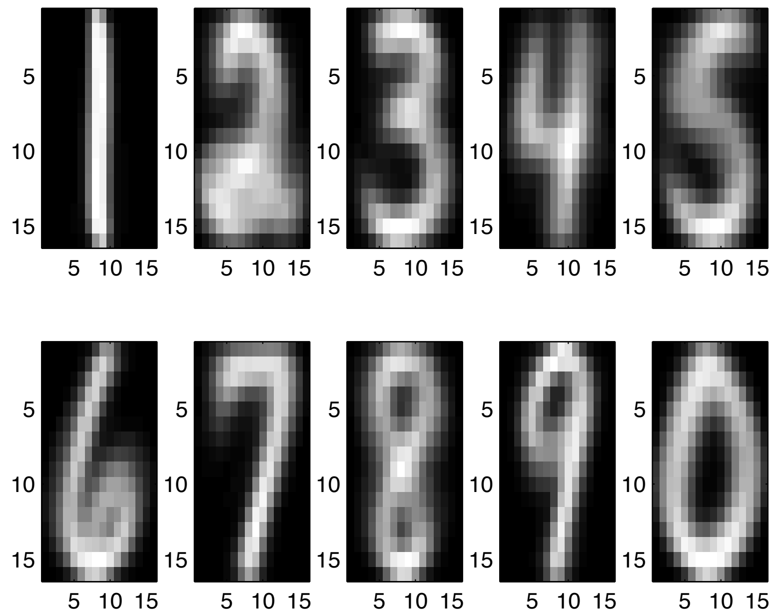


Partie 2.2

```
clc
clear all
```

Q2. chiffres moyens

```
m = size(x, 2);
Xmean = zeros(10, m);

for i=1:10
    Xmean(i,:) = mean(x(y==i, :));
end


figure;
for i=1:10
    subplot(2,5,i);
    imagesc(reshape(Xmean(i,:),16,16)');
    colormap gray;
end
```

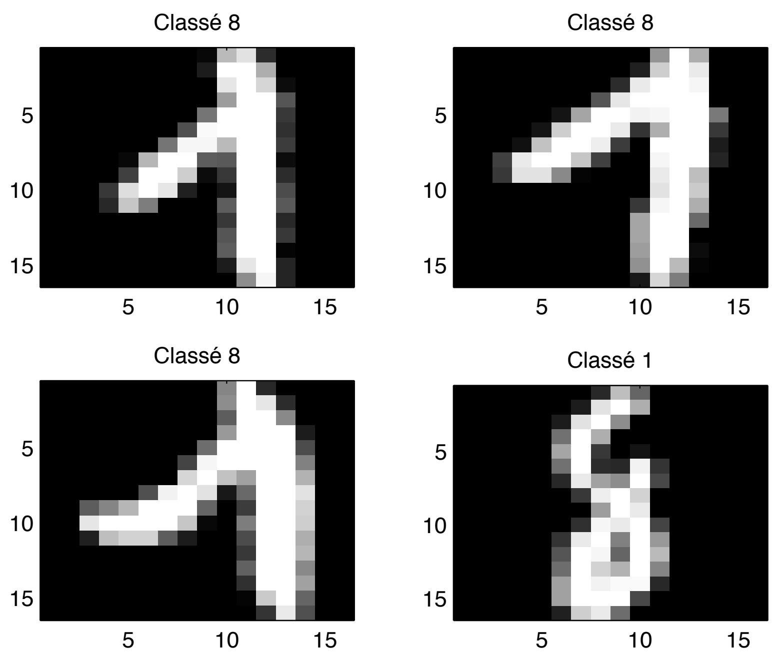


Q3. Discrimination des 1 et des 8

```
X = [x(y==1, :); x(y==8, :)];
Y = [y(y==1); y(y==8)];
Y(Y == 8) = -1;

[n, m] = size(X);

C = 0.001;

cvx_begin
    variable w(m)
    variable b(1)
    variable ksi(n)

    minimize(1/2 * w'*w + C * sum(ksi) )

    subject to

        Y .* (X * w + b) >= 1 - ksi;
        ksi >= 0;

cvx_end

yReconst = monsvmval(X, w, b);
nbErreurs = sum(Y ~= yReconst)
```

```

if (nbErreurs > 0)
    Yerr = yReconst(Y ~= yReconst);
    Yerr(Yerr == -1) = 8;
    Xerr = X(Y ~= yReconst, :);

    for k=1:min(4,size(Xerr,1))
        subplot(2,2,k);
        imagesc(reshape(Xerr(k,:),16,16)');
        title(['Classé ' int2str(Yerr(k))]);
    end
end
```

```
nbErreurs =

     4
```



Errors

Q5. Conclusion

Il n'y a quasiment aucune erreur. On obtient une erreur uniquement a partir de C < 0.001.