Data Mining
TP6 - Théorie bayésienne de la décision

Rémi MUSSARD - Thomas ROBERT

1 Question 1

Préparation des données

```
% taille des échantillons
n1 = 50;
n2 = n1;
n = n1 + n2;

% paramètres lois
mu1 = [0 0];
mu2 = [2 2];
mu = [mu1; mu2];
S = [1 0.5 ; 0.5 4];

% probas à priori
p = [n1/n, n2/n];

% échantillons aléatoires
X1 = randn(n1,2)*S^(1/2) + repmat(mu1, n1, 1);
X2 = randn(n2,2)*S^(1/2) + repmat(mu2, n2, 1);

% données
X = [X1;X2];
```

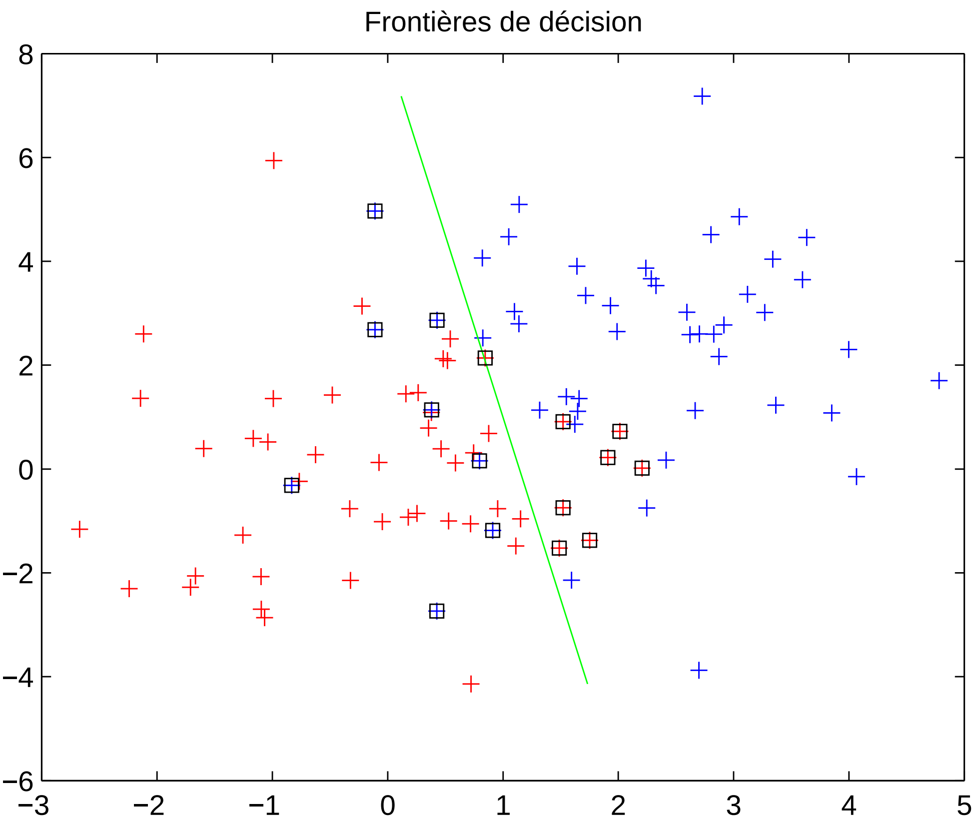
1.a. Frontière de décision théorique

On calcule et un trace la frontière de décision théorique d'après les moyennes et variances utilisées pour générer les points. On trace cette frontière.

```
% affichage des points
close all;
plot(X1(:,1), X1(:,2), '+r');
hold on
plot(X2(:,1), X2(:,2), '+b');
title('Frontières de décision');

% frontière de décision d'après la formule du cours (la 2ème partie de la
% formule disparait car les probas à priori sont identiques)
w = S\(mu1'-mu2');
x0 = (mu1+mu2)/2;

y = max(X(:,2));
x = x0(1) + w(2)*(x0(2) - y)/w(1);
y2 = min(X(:,2));
x2 = x0(1) + w(2)*(x0(2) - y2)/w(1);

plot([x;x2],[y;y2], '-g');
```

```

% nombre d'erreurs
inds1 = find(w'*(X1-repmat(x0,n1,1))'<0);
inds2 = find(w'*(X2-repmat(x0,n1,1))'>0);
erreursX1 = length(inds1);
erreursX2 = length(inds2);
erreursTheo = erreursX1 + erreursX2

plot([X1(inds1,1);X2(inds2,1)],[X1(inds1,2);X2(inds2,2)], 'sk');
```

```
erreursTheo = 16
```



1.b. estimation des moyennes et matrice de covariance

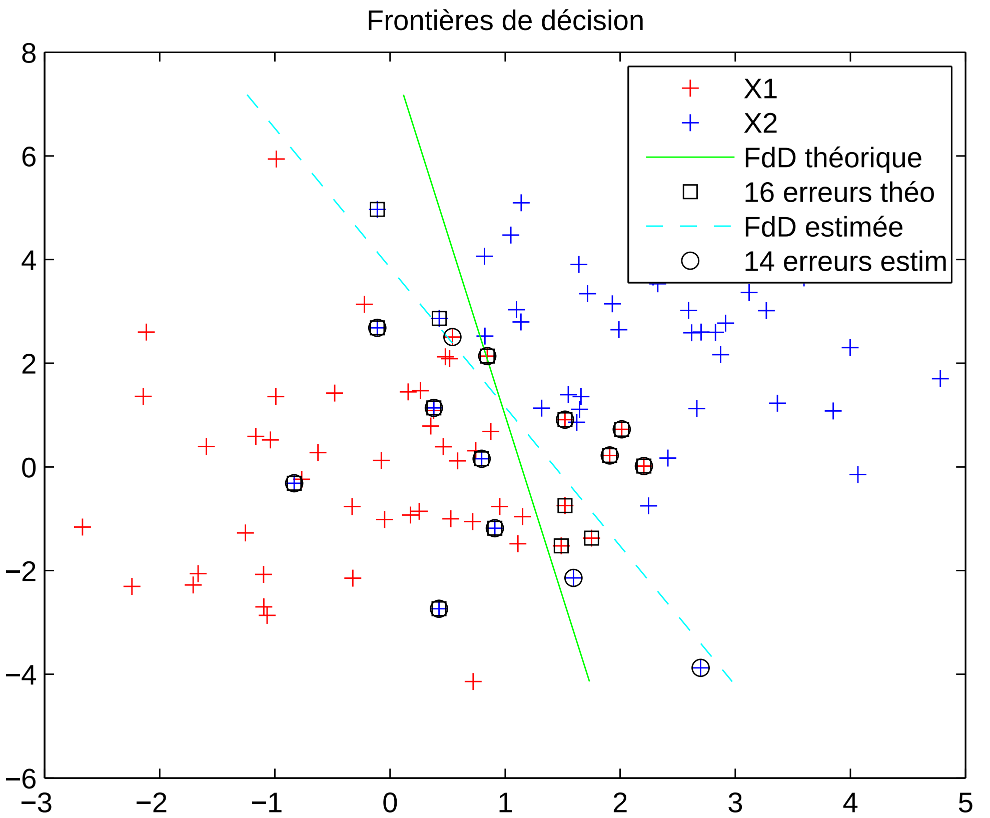
On estime les paramètres selon le maximum de vraisemblance et on utilise les paramètres estimés pour calculer une frontière de décision estimée.

Sur les données d'apprentissage, on fait moins d'erreur en moyenne avec la frontière de décision estimée qu'avec la frontière théorique. Cependant, sur des données de tests, il est probable que ce soit l'inverse puisque la frontière théorique est la frontière idéale pour un jeu de données de taille infinie.

```
% estimation
mu1hat = mean(X1)
mu2hat = mean(X2)
Shat = (cov(X1) + cov(X2)) / 2

% frontière de décision estimée
w = Shat\(mu1hat'-mu2hat');
x0 = (mu1hat+mu2hat)/2;
```

```

y = max(X(:,2));
x = x0(1) + w(2)*(x0(2) - y)/w(1);
y2 = min(X(:,2));
x2 = x0(1) + w(2)*(x0(2) - y2)/w(1);

plot([x;x2],[y;y2], '--c');

% nombre d'erreurs
inds1 = find(w'*(X1-repmat(x0,n1,1))'<0);
inds2 = find(w'*(X2-repmat(x0,n1,1))'>0);
erreursX1 = length(inds1);
erreursX2 = length(inds2);
erreursEstim = erreursX1 + erreursX2

plot([X1(inds1,1);X2(inds2,1)],[X1(inds1,2);X2(inds2,2)], 'ok');

% légende
legend('X1','X2', ...
    'FdD théorique', [num2str(erreursTheo) ' erreurs théo'],...
    'FdD estimée', [num2str(erreursEstim) ' erreurs estim']);
```

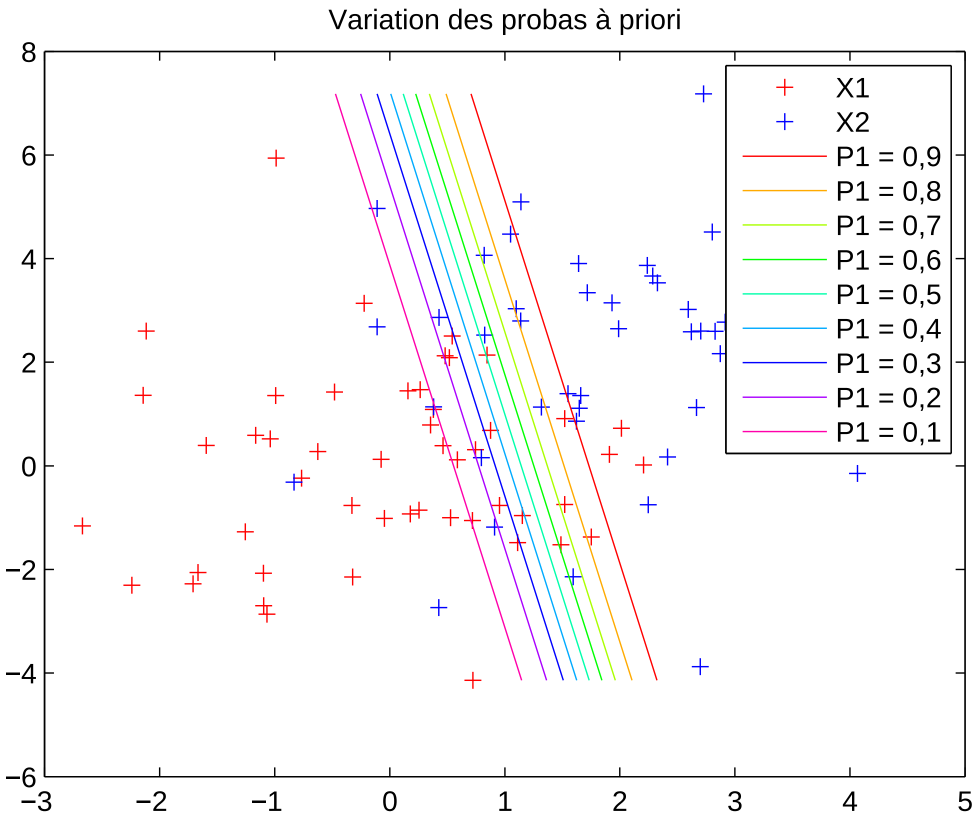
```
mu1hat =   -0.0293   -0.0191

mu2hat =    2.0732    2.2202

Shat =      1.4714    0.2050
            0.2050    3.8767

erreursEstim = 14
```



1.c. Variation des probabilités à priori

On décide de faire varier les probabilités à priori afin de voir l'impact que cela à.

```
figure;
plot(X1(:,1), X1(:,2), '+r');
hold on
plot(X2(:,1), X2(:,2), '+b');
title('Variation des probas à priori');

w=S\(mu1'-mu2');

P1list = 0.1:0.1:0.9;
colors = hsv(9);

for i=1:length(P1list)
    P1 = P1list(i);
    P2 = 1 - P1;

    x0 = 1/2*(mu1'+mu2')'-repmat(log(P1/P2)/((mu1'-mu2')'*w),1,2);
    y = max(X(:,2));
    x = x0(1) + w(2)*(x0(2) - y)/w(1);
    y2 = min(X(:,2));
    x2 = x0(1) + w(2)*(x0(2) - y2)/w(1);

    plot([x;x2],[y;y2], '-', 'Color', colors(i,:));
end

legend('X1', 'X2', ...
    'P1 = 0,9',...
    'P1 = 0,8',...
    'P1 = 0,7',...
    'P1 = 0,6',...
    'P1 = 0,5',...
    'P1 = 0,4',...
    'P1 = 0,3',...
    'P1 = 0,2',...
    'P1 = 0,1');
```



On remarque que (logiquement) cela déplace la frontière de décision vers le cluster ayant la plus grande probabilité à priori.

1.d. Rejet si ambiguïté

On a remarqué que les clusters étaient très proches l'un de l'autre. On décide donc de rejeter les points trop proche de la frontière de décision afin de réduire le nombre d'erreurs de classification.

Après 500 itérations pour chaque valeur de α, on calcule le nombre moyen de points rejetés, le nombre de rejets inutile (nombre de points rejetés alors qu'ils auraient bien été catégorisés), et les taux de classifications erronés avec et sans rejet.

On obtient les résultats suivants :

α	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Nombre de rejets moyen	78	73	70	66	63	61	58	53	54	0
Rejets inutiles	65	49	39	31	25	19	14	9	5	0
Taux d'erreur avant rejet	15%	15%	15%	15%	15%	15%	15%	15%	15%	15%
Taux d'erreur après rejet	2%	4%	5%	6%	8%	9%	11%	12%	13%	15%

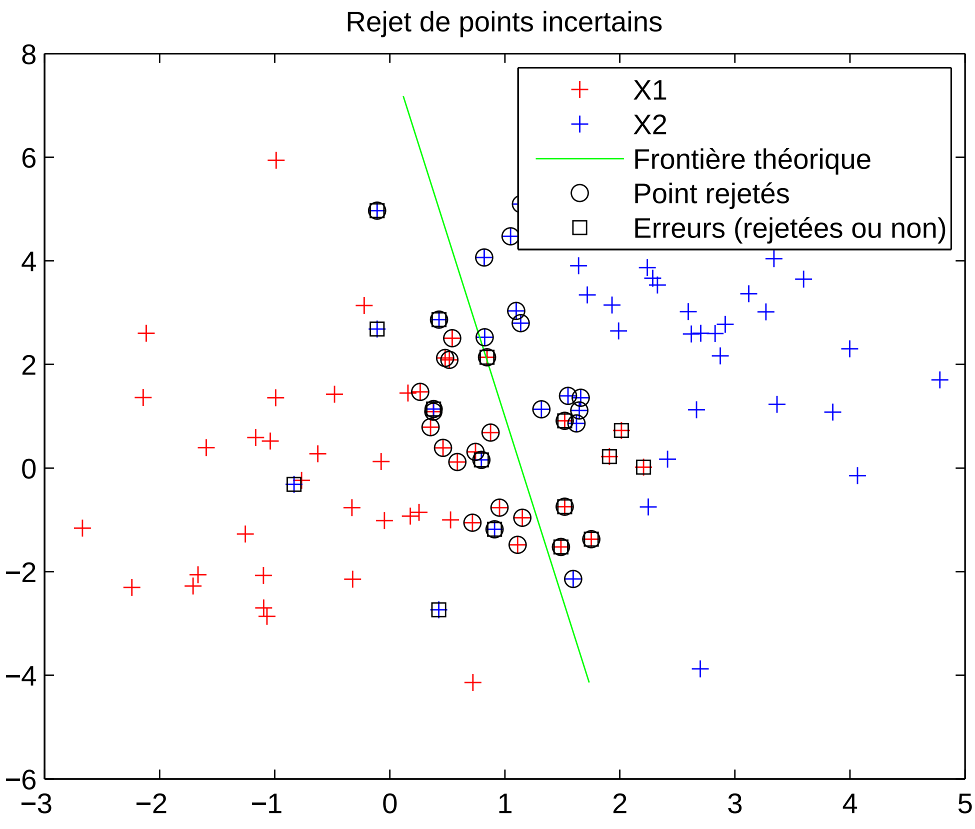
Le choix d'un α dépendra est donc un compromis entre le fait de rejeter beaucoup de point et le fait de conserver des erreurs. Un α de $0,20$ nous semble un bon compromis.

Note : le code en italique permet de générer les résultats de tests des différentes valeurs de α visibles ci-dessus.

```
% seuil de rejet
alpha = 0.2

% statistiques
% stats = [];
% for alpha = 0.05:0.05:0.50
% rejetsInutile = [];
```

```
% nbsErreursAvantRejet = [];
% nbsErreursApresRejet = [];
% nbsRejets = [];
% for i = 1:500
% X1 = randn(n1,2)*S^(1/2) + repmat(mu1, n1, 1);
% X2 = randn(n2,2)*S^(1/2) + repmat(mu2, n2, 1);
% X = [X1;X2];

% probas à priori
P_1 = n1/n;
P_2 = n2/n;

% probabilités conditionelles
P_x_1 = mvnpdf(X, mu1, S);
P_x_2 = mvnpdf(X, mu2, S);

% loi marginale de X
P_x = P_1 * P_x_1 + P_2 * P_x_2;

% probabilités à posteriori
P_1_x = (P_x_1 * P_1)./P_x;
P_2_x = 1-P_1_x;

% affichage
figure;
plot(X1(:,1), X1(:,2), '+r');
hold on
plot(X2(:,1), X2(:,2), '+b');
title('Rejet de points incertains');

% frontière de décision
w = S\(mu1'-mu2');
x0 = (mu1+mu2)/2;

y = max(X(:,2));
x = x0(1) + w(2)*(x0(2) - y)/w(1);
y2 = min(X(:,2));
x2 = x0(1) + w(2)*(x0(2) - y2)/w(1);

plot([x;x2],[y;y2], '-g');

% indices des points à rejeter
inds = ((P_1_x - P_2_x < 0) | (P_1_x < 1 - alpha)) & ...
       ((P_2_x - P_1_x < 0) | (P_2_x < 1 - alpha));

% affichage des points à rejeter
plot(X(inds,1), X(inds,2), 'ok');

% nombre d'erreurs
inds1 = (w'*(X1-repmat(x0,n1,1))'<0);
inds2 = (w'*(X2-repmat(x0,n1,1))'>0);
inds1avecRejet = inds1' & ~inds(1:length(X1));
inds2avecRejet = inds2' & ~inds(length(X1)+1:end);

% affichage
plot([X1(inds1,1);X2(inds2,1)],[X1(inds1,2);X2(inds2,2)], 'sk');

legend('X1', 'X2', 'Frontière théorique', 'Point rejetés', 'Erreurs (rejetées ou non)');

% statistiques
nbRejets = sum(inds);
nbErreursAvantRejet = sum(inds1) + sum(inds2);
erreursX1 = sum(inds1avecRejet);
erreursX2 = sum(inds2avecRejet);
nbErreursApresRejet = erreursX1 + erreursX2;
nbRejetsUtiles = nbErreursAvantRejet - nbErreursApresRejet;
nbRejetsInutiles = nbRejets - nbRejetsUtiles;
if (nbRejets > 0)
    rejetInutile = nbRejetsInutiles / nbRejets * 100;
else
    rejetInutile = 0;
```

```
end

% statistiques
% nbsErreursAvantRejet = [nbsErreursAvantRejet nbErreursAvantRejet];
% nbsErreursApresRejet = [nbsErreursApresRejet nbErreursApresRejet];
% nbsRejets = [nbsRejets nbRejets];
% rejetsInutile = [rejetsInutile rejetInutile];
% end
% stats = [stats [alpha ;
%     mean(rejetsInutile);
%     mean(nbsRejets);
%     mean(nbsErreursAvantRejet)/n*100;
%     mean(nbsErreursApresRejet)/(n-mean(nbsRejets))*100;
%     ]];
% end
%
```



2 Question 2

2.a Chargement des données

Partage des données en données d'apprentissage et données de test grâce à la fonction `splitdata` disponible sur Moodle.

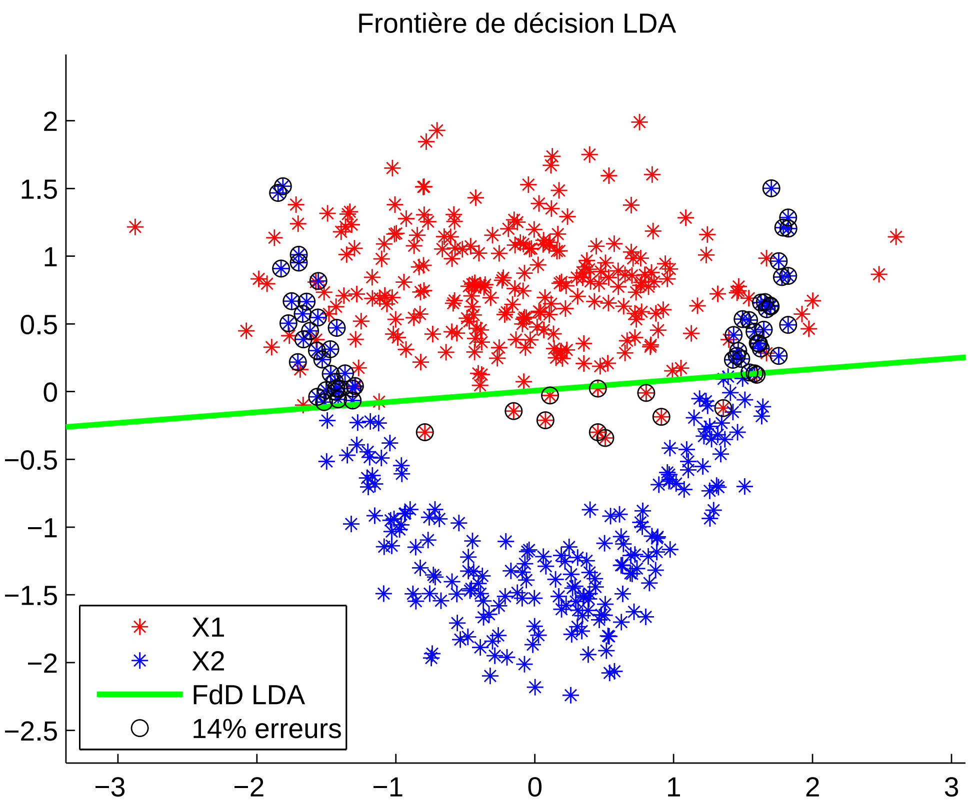
```
load('clownsv7.mat');

[xApp, yApp, xTest, yTest] = splitdata(X, y, 0.5);
```

2.b. et 2.c. Frontières et erreurs en LDA

On calcule la frontière de décisions pour la LDA et on calcule le nombre d'erreurs de classification.

```
x1 = xApp(yApp > 0, :);
x2 = xApp(yApp < 0, :);
x12 = [x1; x2];

mu1 = mean(x1);
mu2 = mean(x2);

S = (cov(x1) + cov(x2)) / 2;

% calcul de la frontière de décision en LDA

w = S\(mu1'-mu2');
x0 = (mu1+mu2)/2;

Y1 = max(x12(:,2));
X1 = x0(1) + w(2)*(x0(2) - Y1)/w(1);
Y2 = min(x12(:,2));
X2 = x0(1) + w(2)*(x0(2) - Y2)/w(1);

% Erreurs LDA
[n1, m1] = size(x1);
[n2, m2] = size(x2);

inds1 = find(w'*(x1-repmat(x0,n1,1))'<0);
inds2 = find(w'*(x2-repmat(x0,n2,1))'>0);
erreursLDA = (length(inds1) + length(inds2))
ratioErreursLDA = erreursLDA/length(x12)

% affichage
figure; hold on;
plot(x1(:,1), x1(:,2), '*r');
plot(x2(:,1), x2(:,2), '*b');
plot([X1;X2],[Y1;Y2], '-g', 'linewidth', 2);
plot([x1(inds1,1) ; x2(inds2,1)], [x1(inds1,2) ; x2(inds2,2)], 'ok');
axis([min(x12(:,1))-0.5 max(x12(:,1))+0.5 min(x12(:,2))-0.5 max(x12(:,2)+0.5)]);
title('Frontière de décision LDA')
leg = legend('X1', 'X2', 'FdD LDA', ...
    [int2str(round(ratioErreursLDA*100)) '% erreurs']);
set(leg,'Location','SouthWest')
```

```
erreursLDA = 70
ratioErreursLDA = 14%
```



A la vue des données, tracer une frontière de décision via une LDA ne semble pas convenir. En effet, la frontière entre les données ne semble pas linéaire. Il serait plus judicieux de faire une frontière de décision via une QDA, étant donné le caractère hyperbolique apparent de la frontière.

Lorsque l'on calcule l'erreur que nous avons avec la LDA, on obtient un environ 14% d'erreurs.

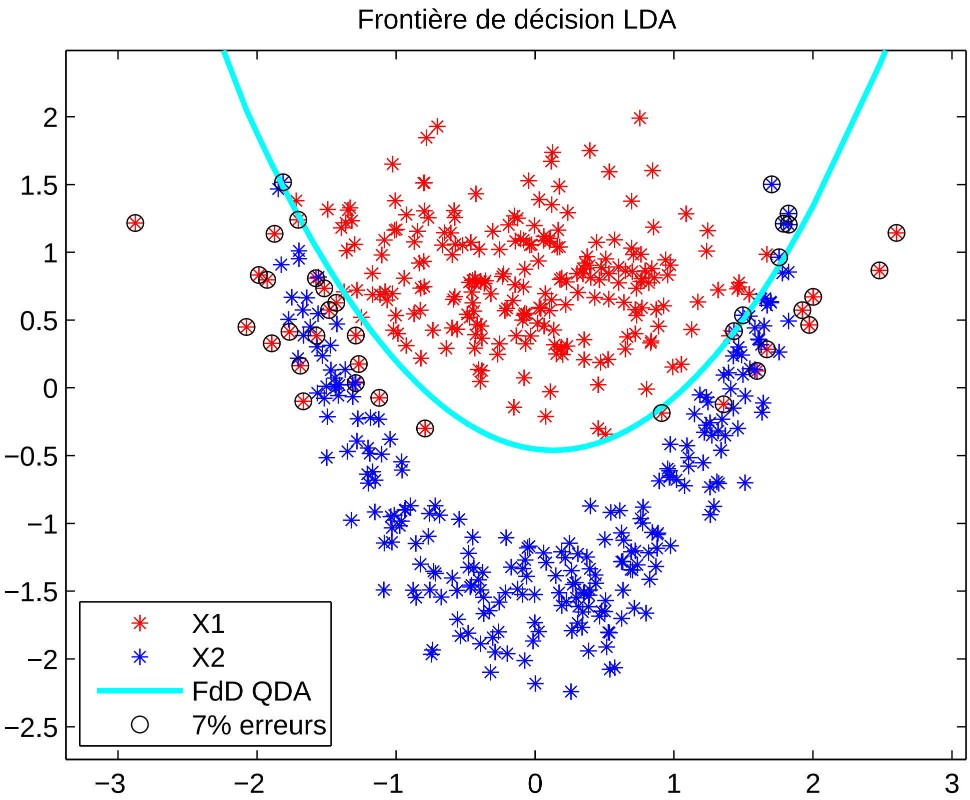
2.b. et 2.c. Frontières et erreurs en QDA

On calcule maintenant la frontière de décisions pour la QDA et on calcule le nombre d'erreurs de classification pour cette nouvelle méthode.

```
% calcul de la QDA
Wj = inv(S)/2;
wj = 2*Wj*mu1';
wj0 = (mu1*wj)/2 - (log(norm(2*Wj)))/2 + log(length(x1)/length(xApp));

xQDA = sort(x12(:,1));
yQDA = wj0 + wj(1)*xQDA + Wj(1,1)*xQDA.^2;

% Erreurs QDA
x = x1(:,1);
yQDA1 = wj0+wj(1)*x+Wj(1,1)*x.^2;
x = x2(:,1);
yQDA2 = wj0+wj(1)*x+Wj(1,1)*x.^2;

inds1 = find((x1(:,2)-yQDA1)<0);
inds2 = find((x2(:,2)-yQDA2)>0);
ratioErreursQDA = (length(inds1) + length(inds2))/length(x12)

% affichage
figure;
plot(x1(:,1), x1(:,2), '*r');
hold on
plot(x2(:,1), x2(:,2), '*b');
```

```
plot(xQDA,yQDA,'-c', 'linewidth', 2);
plot([x1(inds1,1) ; x2(inds2,1)], [x1(inds1,2) ; x2(inds2,2)], 'ok');
axis([min(x12(:,1))-0.5 max(x12(:,1))+0.5 min(x12(:,2))-0.5 max(x12(:,2)+0.5)]);
title('Frontière de décision LDA')
leg = legend('X1', 'X2', 'FdD QDA', ...
    [int2str(round(ratioErreursQDA*100)) '% erreurs']);
set(leg,'Location','SouthWest')
```

```
ratioErreursQDA = 7,4 %
```



La frontière de décision réalisé avec la QDA semble plus appropriée. On obtient une frontière de décision de forme hyperbolique qui sépare relativement bien les données en deux classes.

En calculant l'erreur de classification sur QDA, on obtient un taux d'erreur d'environ 8%, ce qui est moins élevé que le taux d'erreur de la LDA.

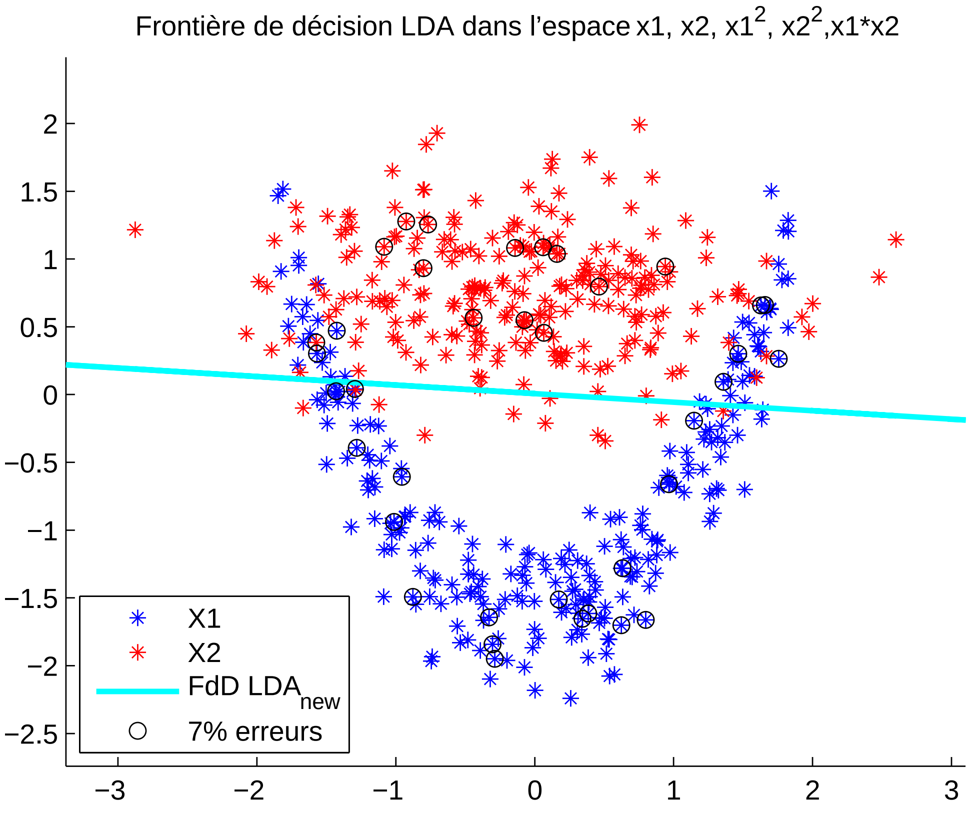
2.d. Frontière de décision pour LDA dans l'espace $\{x_1, x_2, x_1^2, x_2^2, x_1.x_2\}$

On trace cette fois la frontière LDA dans l'espace $\{x_1, x_2, x_1^2, x_2^2, x_1.x_2\}$.

```
x1carre = x12(:,1).*x12(:,1);
x2carre = x12(:,2).*x12(:,2);
x1x2 = x12(:,1).*x12(:,2);

xfinal = [x12 x1carre x2carre x1x2];
xnew1 = xfinal(yApp > 0, :);
xnew2 = xfinal(yApp < 0, :);

```

```
% etimation paramètres
mu1 = mean(xnew1);
mu2 = mean(xnew2);
S = (cov(xnew1) + cov(xnew2)) / 2;

% calcul FdD
w = S\(mu1'-mu2');
x0 = (mu1+mu2)/2;

X1 = min(xfinal);
X2 = max(xfinal);
Y1 = x0(2) - (w(1)*(x0(1)-X1))/w(2);
Y2 = x0(2) - (w(1)*(x0(1)-X2))/w(2);

% erreurs
inds1 = find(w'*(xnew1 - repmat(x0, size(xnew1,1), 1))' < 0);
inds2 = find(w'*(xnew2 - repmat(x0, size(xnew2,1), 1))' > 0);
ratioErreursLDAnew = (length(inds1) + length(inds2))/length(xfinal)

% affichage
figure; hold on;
plot(xnew1(:,1),xnew1(:,2), '*b');
plot(xnew2(:,1),xnew2(:,2), '*r');
plot([X1 X2],[Y1 Y2], '-c', 'linewidth', 2);
plot([x1(inds1,1) ; x2(inds2,1)], [x1(inds1,2) ; x2(inds2,2)], 'ok');
axis([min(x12(:,1))-0.5 max(x12(:,1))+0.5 min(x12(:,2))-0.5 max(x12(:,2)+0.5)]);
title('Frontière de décision LDA dans l''espace {x1, x2, x1^2, x2^2,x1*x2}')
leg = legend('X1', 'X2', 'FdD LDA_{new}', ...
    [int2str(round(ratioErreursLDAnew*100)) '% erreurs']);
set(leg,'Location','SouthWest')
```

```
ratioErreursLDAnew = 7.4 %
```



Il est donc impossible de visualiser cette frontière dans cet espace de $\mathbb{R}^5$, on se contente donc de le visualiser sur les deux premières composantes de cet espace. Cependant, il est important de noter qu'il ne s'agit que donc que d'une partie des composantes des points et que la frontière est en fait un hyperplan. On ne peut donc pas visualiser quels points sont de tel ou tel côté de la frontière directement. Ces points en erreur ont tout de même été marqués.

Le taux d'erreur est ici d'environ 8%.

2.e. Comparaison des résultats

Appliquer la LDA dans un espace de plus grande dimension permet d'améliorer la précision de la LDA. On se retrouve ici avec un taux d'erreur très proche de celui de la QDA réalisée précédemment.

Cependant, le fait de devoir augmenter le nombre de dimensions de la LDA la rend sans doute plus gourmande en ressource qu'une QDA.