Rapport de travaux pratique MA412

TP3 : Discrimination

Kot Michel - Mathieu MARLEIX - Élève d'I4 - ESIEE Paris

Table des matières

Introduction

Ce TP à pour but l'utilisation des différents type de discrimination vu en cours.

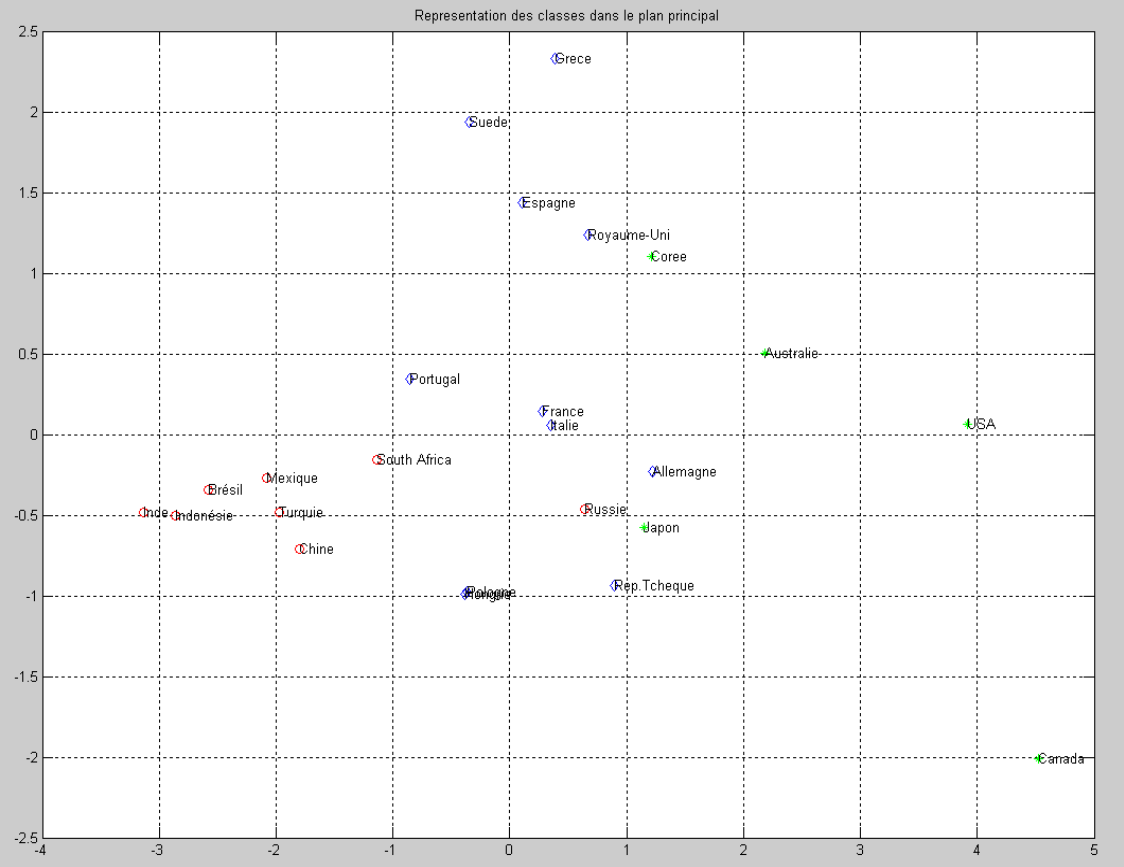
I. Représentation des données

I. 1.. Normalisation des données

```
% Normalisation et visualisation des données
classr(X,L,pays)
```

On utilise la fonction normtab modifiée pour normaliser les données : On utilise Xc à laquelle on soustrait la moyenne et l'écart-type de X en prenant en compte la taille de Xc.

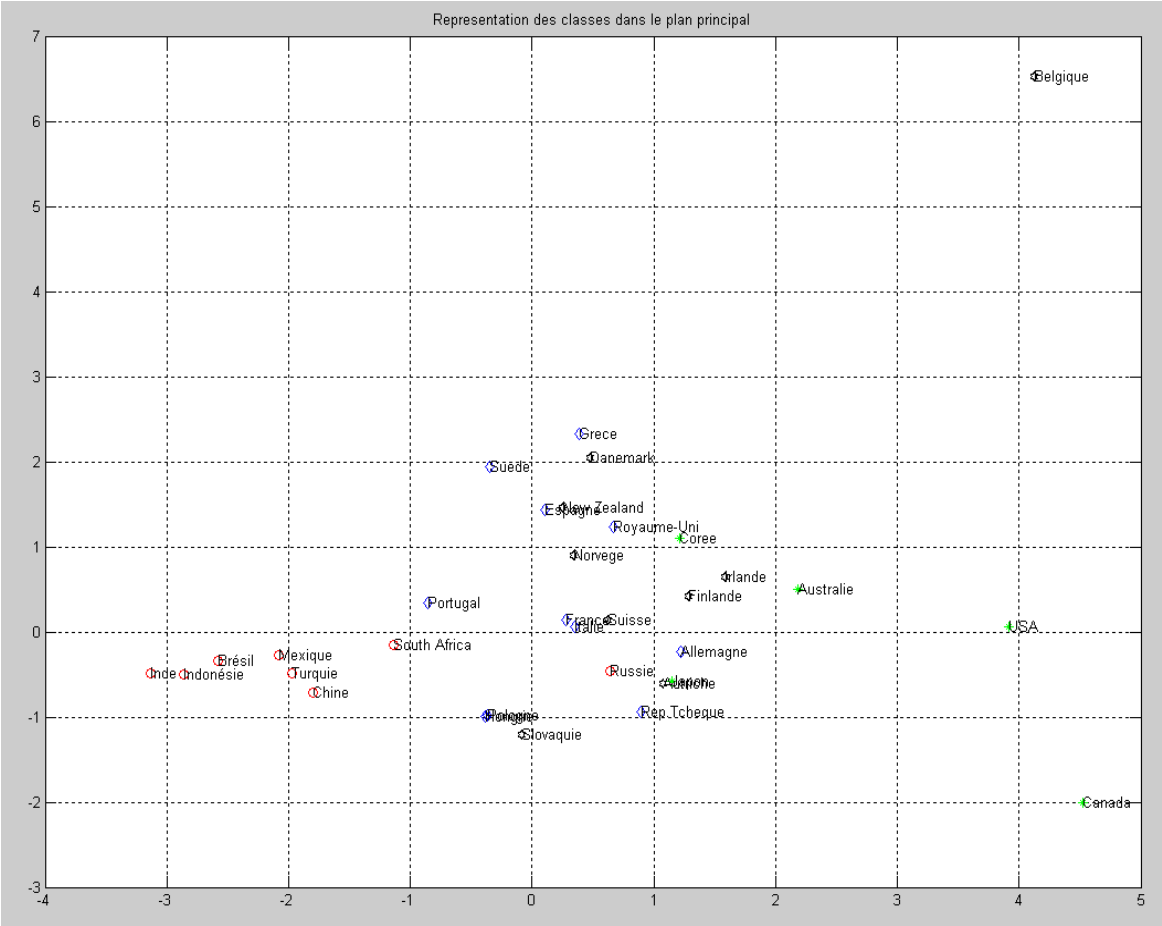
```
% Normalisation et projection de Xc
Xnc=(Xc-repmat(mean(X),size(Xc,1),1))*inv(diag(std(X,1)'));
```



II. 2. Représentation dans le plan principal

Pour pouvoir représenter les nouveaux pays sur le plan existant, on modifie la fonction classr. Cette fonction se voit passer en argument en plus de ses arguments de bases, les données normalisées et le texte des nouveaux pays. On modifie la partie de représentation et on trace avec un hold on sur la figure existante. (Voir Annexe 1 pour le code de classr2).

```
figure
classr2(X,Xnc,L,pays,paysc)
```



II. Discrimination par règle des k-ppv

Les options de la fonction knclassify nous permettent de spécifier la règle de discrimination :

- nearest : Règle majorité avec en cas d'égalité le plus proche voisin est choisit
- random : Règle majorité avec en cas d'égalité un point au hasard est choisit
- consensus : Règle de consensus, tout les plus proches voisins doivent être en accord.

Commandes :

```
% Application de la fonction knnclassify
class1n = knnclassify(Xnc,X,L,1,'euclidean','nearest')
class2n = knnclassify(Xnc,X,L,2,'euclidean','nearest')
class3n = knnclassify(Xnc,X,L,3,'euclidean','nearest')
class4n = knnclassify(Xnc,X,L,4,'euclidean','nearest')
class1r = knnclassify(Xnc,Xn,L,1,'euclidean','random')
class2r = knnclassify(Xnc,Xn,L,2,'euclidean','random')
class3r = knnclassify(Xnc,Xn,L,3,'euclidean','random')
class4r = knnclassify(Xnc,Xn,L,4,'euclidean','random')
class1c = knnclassify(Xnc,Xn,L,1,'euclidean','consensus')
class2c = knnclassify(Xnc,Xn,L,2,'euclidean','consensus')
class3c = knnclassify(Xnc,Xn,L,3,'euclidean','consensus')
class4c = knnclassify(Xnc,Xn,L,4,'euclidean','consensus')
```

Résultats :

```
class1n =   1   1   1   1   3   1   1   1   1

class2n =   1   1   1   1   3   1   1   1   1

class3n =   1   1   1   1   3   1   1   1   1

class4n =   1   1   1   1   3   1   1   1   1

class1r =   1   1   1   1   3   1   1   1   1

class2r =   1   1   1   1   3   1   1   1   1

class3r =   1   1   1   1   3   1   1   1   1

class4r =   1   1   1   1   1   1   1   1   1

class1c =   1   1   1   1   3   1   1   1   1

class2c =   1   1  NaN   1   3   1   1   1   1

class3c =   1   1  NaN   1  NaN   1   1  NaN   1

class4c =  NaN  NaN  NaN   1  NaN   1  NaN  NaN   1
```

On observe ici que les point son toujours bien déterminés sauf pour la méthode consensus posant des problèmes quand les plus proches voisins ne sont pas en accord. Les points peuvent changer de classe en fonction de la règle utilisée.

III. Discrimination Bayésienne

La fonction classify nous permet de réaliser une discrimination des données étudiées (Bayésienne ou Mahalanobis). En observant la documentation de classify (help classify), la documentation nous renseigne sur les différentes méthodes de séparation disponible : Linear séparation linéaire, Quadratic effectue une séparation quadratique et Mahalanobis effectuant une spéaration à l'aide de la distance de mahalanobis.

> Linear discrimination fits a multivariate normal density to each group, with a pooled estimate of covariance.
> Quadratic discrimination fits MVN densities with covariance estimates stratified by group.
> Mahalanobis discrimination uses Mahalanobis distances with stratified covariance estimates.

On applique cette fonction aux données normalisées.

```
% Application de la fonction classify
[classl,Pl,errl] = classify(Xnc,Xn,L,'linear')
[classq,Pq,errq] = classify(Xnc,Xn,L,'quadratic')
[classm,Pm,errm] = classify(Xnc,Xn,L,'mahalanobis')
```

```
Classl =  3    1    3    1    3    1    3    1    1

ERR =  0.0720

POST =  0.1307    0.0015    0.8678
        0.9769    0.0226    0.0005
        0.2585    0.0000    0.7415
        0.9160    0.0011    0.0829
        0.1102    0.0006    0.8892
        0.9976    0.0002    0.0021
        0.1332    0.0018    0.8649
        0.5263    0.4727    0.0010
      0.9990    0.0010    0.0000

??? Error using ==> classify at 299
The covariance matrix of each group in TRAINING must be positive definite.

Error in ==> tp3 at 53
classq = classify(Xnc,Xn,L,'quadratic')
```

Avec ERR : taux d'erreur , POST : probabilité a posteriori d’affectation.

On constate que le calcul à l'aide de la méthode linéaire renvoi un bon résultat, mais que des erreurs sont perçues sur les deux autres méthodes : la méthode de calcul par la distance quadratique à besoin que la matrice soit être nécessairement définie positive. De plus la distance de mahalanobis utilise la distance quadratique.

On utilise les données normalisées projetées.

IV. Annexes

Code de classr2

```
function [Xncp] = classr2(X,Xnc,L,q,paysc)

[n,p]=size(X);
k=max(L);
no=num2str([1:n]');
if p>2
%Analyse en composantes principales
Xn=normtab(X);
[V,D]=eig(corrcoef(X));
[Y I]=sort(-diag(D));
V=V(:,I);
Vf=V*sqrt(-Y);
Vc=Xn*V;
X=Vc(:,[1 2]);
Vcc=Xnc*V;
Xncp=Vcc(:,[1 2]);

end

%Dénomination souhaitée
if nargin<3
etiq=no;
else
etiq=q;
end

%Repr?sentation dans le plan si k<=6
if k<=7
plot(X((L==1),1),X((L==1),2),'bd')
hold on
plot(X((L==2),1),X((L==2),2),'ro')
plot(X((L==3),1),X((L==3),2),'g*')
plot(X((L==4),1),X((L==4),2),'ys')
plot(X((L==5),1),X((L==5),2),'m+')
plot(X((L==6),1),X((L==6),2),'cx')
plot(X((L==7),1),X((L==7),2),'kh')
text(X(:,1),X(:,2),etiq)
hold off
title('Representation des classes dans le plan principal')
grid
end

%
hold on;
plot(Xncp(:,1),Xncp(:,2),'kh')
```

```
text(Xncp(:,1),Xncp(:,2),paysc)
hold off


function [Xn]=normtab(X)

%[Xn]=NORMTAB(X)
%Normalisation du tableau de donnees X

Xn=(X-repmat(mean(X),size(X,1),1))*inv(diag(std(X,1)'));
```