Reconnaissance des formes et inspection, Sys800

Laboratoire 1

Représentation, extraction et sélection de caractéristiques

Un système de reconnaissance de formes a pour objectif de prédire la classe d'appartenance d'une forme inconnue. Pour y parvenir, trois étapes sont nécessaires. La première étape dite de prétraitement consiste à isoler la forme à reconnaître. La seconde étape consiste ensuite à effectuer un certain nombre de mesures qui vont permettre de caractériser la forme. Le vecteur de caractéristiques alors obtenu sera enfin exploité pour prendre une décision lors de l'étape de classification. Par ailleurs, de manière à optimiser les différents paramètres de chacune de ces étapes, les approches statistiques utilisent un ensemble de données étiquetées que l'on nomme base d'entraînement. Une illustration de cette décomposition du processus de reconnaissance de formes est présentée figure 1.

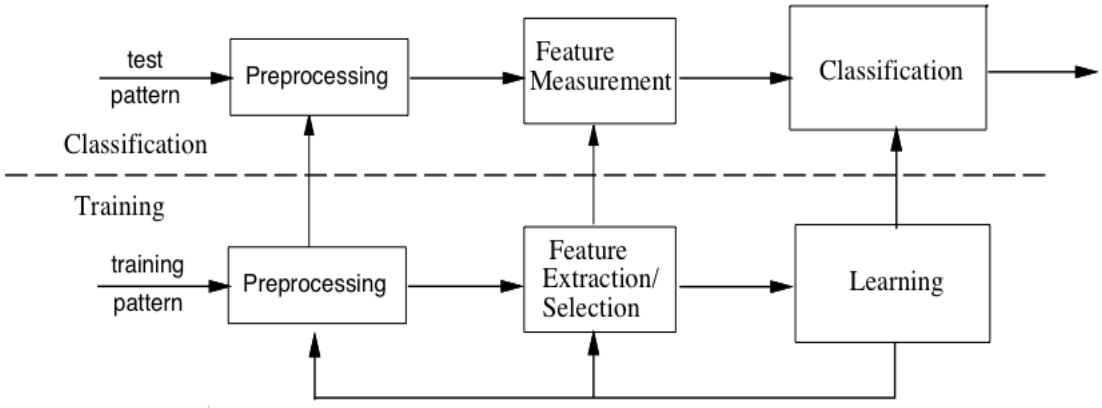


Figure 1 : Approche statistique pour la reconnaissance de formes selon Jain *et al.*[1]

Dans le cadre de ces séances de laboratoire nous nous intéresserons à un « classique » de la reconnaissance de formes : la reconnaissance d'images de chiffres manuscrits.

---

[1] A.K. Jain, R.P.W. Duin, J. Mao, Statistical Pattern Recognition: A Review, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4-37, 2000.

L'étape de prétraitement ayant déjà été réalisée, vous disposez d'une base de données d'images binaires de chiffres isolés, dont quelques exemples sont présentés figure 2. Cette base de données comprend 7000 images et est divisée en deux sous-ensembles. Le premier contient 600 exemples de chacun des chiffres, qui seront utilisés pour entraîner les classifieurs. Le second sous-ensemble contient 100 exemples de chaque classe et sera utilisé pour tester et comparer les différents classifieurs.

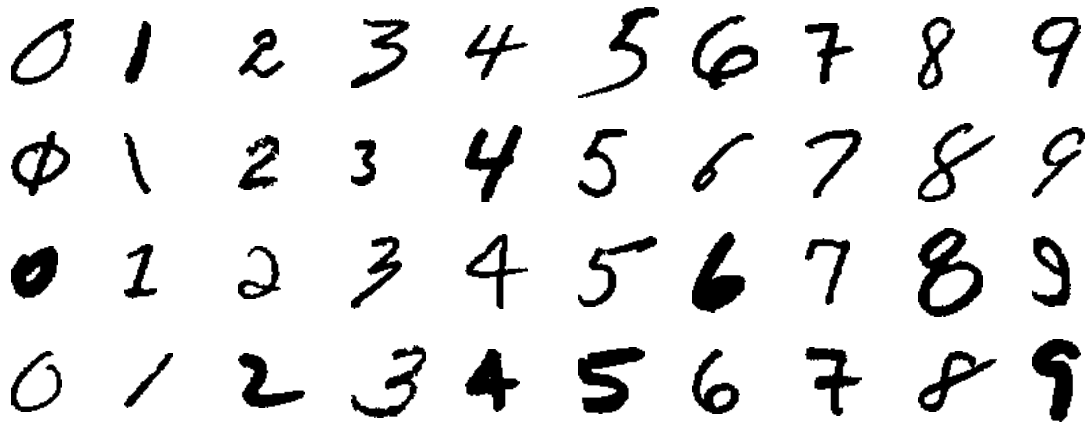


Figure 2 : Quelques exemples d'images de chiffres manuscrits

Ainsi, ce premier laboratoire porte sur l'étape d'extraction de caractéristiques. De nombreuses techniques adaptées à la reconnaissance de l'écriture manuscrite sont décrites dans la littérature[2]. Nous vous proposons d'utiliser pour ce laboratoire le codage rétinien qui est l'une des méthodes les plus simples pour transformer une image binaire de dimensions quelconques en un vecteur de dimension fixe.

[2] O.D. Trier, A.K. Jain, T. Taxt, Feature extraction methods for character recognition - A survey, *Pattern Recognition*, vol. 29, pp. 641-662, 1996.

Malheureusement une représentation brute de l'information conduit souvent à des vecteurs de grandes dimensions. Ainsi, vous chercherez donc à compresser l'information discriminante dans un nombre minimum de caractéristiques de manière à combattre la « malédiction de la dimensionnalité » et à réduire la complexité de calcul de vos algorithmes. Pour ce faire, nous vous proposons d'utiliser l'analyse en composantes principales (ACP), et aussi la sélection de caractéristiques pertinentes selon le critère de Fisher.

En pratique, vous devrez donc dans un premier temps écrire la fonction `retine.m` qui est utilisée au sein du script `make_database.m` fourni en annexe. Ce script vous permettra d'ouvrir une à une toutes les images de la base de données pour en extraire les vecteurs de caractéristiques. Les vecteurs ainsi obtenus seront ensuite stockés ensemble sous forme matriciel puis sauvegardé pour pouvoir être réutilisés par la suite.

Précisons que lorsque vous chargez une image avec la fonction `imread` de Matlab, elle est alors représentée par une matrice dont les éléments à 0 correspondent au caractère et les éléments à 1 au fond blanc. D'autre part, si vous désirez visualiser cette matrice vous pouvez utiliser la fonction `imshow` de Matlab.

La fonction `retine.m` reçoit donc en argument la matrice correspondant à l'image traitée ainsi que les dimensions de la rétine désirée et renverra en sortie un vecteur ligne contenant les valeurs du codage rétinien.

1 – REPRÉSENTATION ET EXTRACTION DE CARACTÉRISTIQUES : LE CODAGE RÉTINIEN

Le codage rétinien consiste à placer une grille virtuelle de dimension fixe sur l'image et à déterminer pour chaque case de la grille le ratio de pixels appartenant au chiffre. Il existe différentes méthodes pour réaliser un codage rétinien. Nous vous proposons une procédure simple et facile à mettre en place.
La figure 3 illustre cette procédure qui peut être décomposée en trois étapes :

- La première étape consiste donc à centrer l'image initiale de dimension [n × m], dans une matrice carrée de dimensions [max(n,m) × max(n,m)].

- Ensuite, de manière à s'assurer que la grille virtuelle tombe juste, nous vous proposons d'agrandir l'image pour que sa dimension devienne [l*max(n,m) × c*max(n,m)], où l représente le nombre de ligne de la rétine et c le nombre de colonne de celle-ci.

- Enfin, il sera alors facile d'extraire de chaque zone la caractéristique correspondant au rapport entre la surface du caractère et la surface de la zone. Il suffira pour cela, de sommer l'ensemble des valeurs des pixels de la zone et de diviser par le nombre total de pixels de la zone. Les différentes valeurs ainsi obtenues seront enfin stockées dans un vecteur de dimension [l*c × 1], qui caractérisera l'image.

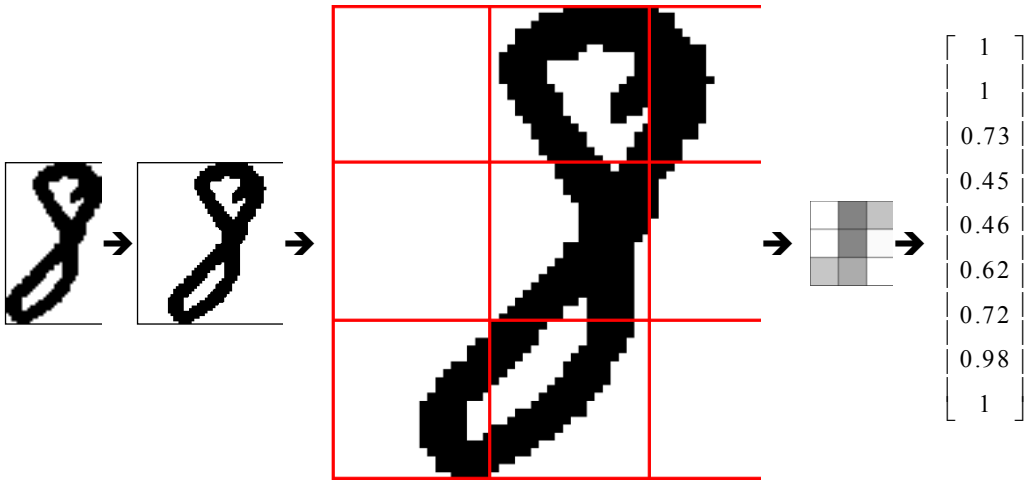
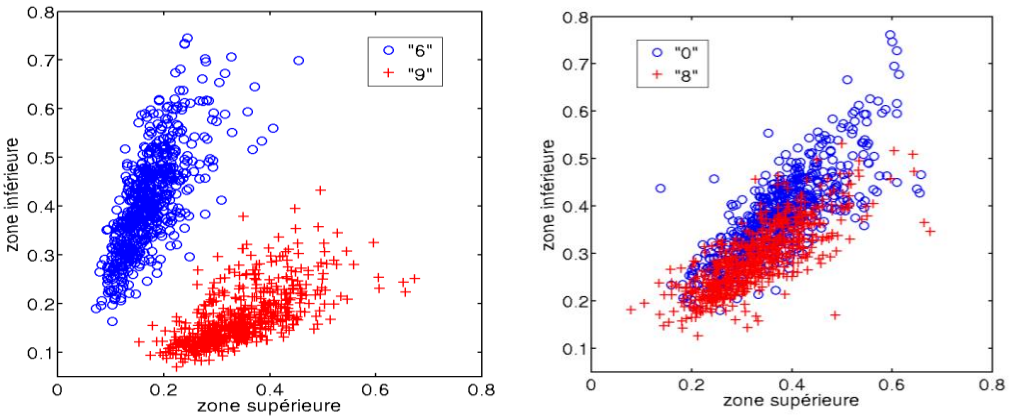


Figure 3 : Illustration de la procédure utilisée pour réaliser le codage rétinien

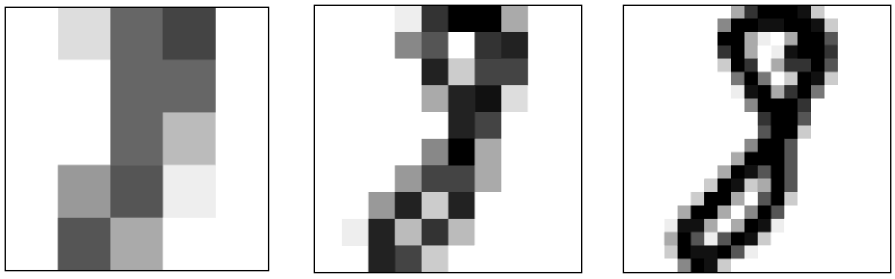
Un exemple de résultat obtenu à l'aide d'une rétine rudimentaire est présenté figure 4. Nous avons alors utilisé une rétine 2×1 de manière à obtenir que deux caractéristiques et pouvoir ainsi visualiser la distribution des données à la manière de ce qui avait été fait lors de la séance précédente de laboratoire.



(a) (b)

Figure 4 : Exemples de résultats obtenus lors de l'utilisation d'une rétine [2×1]

Il est intéressant de constater que ces caractéristiques s'avèrent relativement discriminantes pour résoudre certains sous-problèmes, tels que séparer le chiffre « 6 » du chiffre « 9 » (voir figure 4-a), mais quasiment inutiles pour différentier le « 8 » du « 0 » (voir figure 4-b). Ainsi, la perte d'information est bien entendu trop importante pour résoudre efficacement notre problème de reconnaissance de caractères. Nous pouvons ainsi constater figure 5 que l'utilisation d'une rétine 5×5 permet de réduire fortement la dimension du vecteur de représentation, mais conserve mal l'information visuelle. Par contre, une rétine 20×20 permet une représentation précise des images, mais conduit à des vecteurs de caractéristiques de très grandes dimensions. L'utilisation d'une rétine 10×10 semble alors être un bon compromis entre la qualité de représentation et la dimensionnalité.



5×5 10×10 20×20

Figure 5 : Visualisation des résultats obtenus avec différentes dimensions de rétines

2- RÉDUCTION DE LA DIMENSIONNALITÉ : ACP

L'analyse en composantes principales est une méthode statistique qui consiste à extraire les vecteurs propres et les valeurs propres de la matrice de covariance calculée à partir de l'ensemble de données d'apprentissage. L'objectif de cette technique est donc de rechercher par combinaison linéaire les axes principaux de l'ensemble des données, c'est-à-dire les axes suivant lesquels la variabilité est la plus grande. Ainsi, il est possible de transformer un vecteur de *n* caractéristiques en projetant le point correspondant à ce vecteur sur les *m* axes principaux ($m < n$) ayant les plus fortes valeurs propres. Le vecteur obtenu après projection est donc de dimension *m* et est composé de caractéristiques décorrélées.

La figure 7 illustre le fonctionnement de l'ACP à travers un exemple simple. À partir d'un ensemble de données d'apprentissage décrit par deux caractéristiques x_1 *et* x_2, il est possible de déterminer deux nouvelles caractéristiques y_1 et y_2 formant un repère défini par les vecteurs propres et la moyenne des données. Alors, si l'on considère que l'information discriminante est contenue dans les caractéristiques ayant les plus fortes valeurs propres, il est possible de supprimer la caractéristique y_2 qui peut être considérée comme étant du bruit.

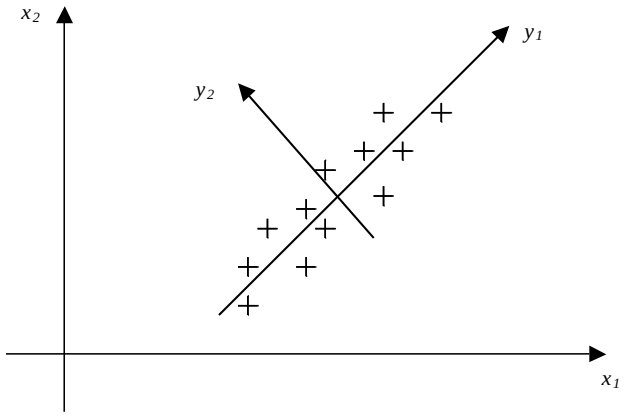


Figure 7 : Illustration de l'ACP à l'aide d'un exemple en 2 dimensions

Si nous reprenons le sous-problème consistant à séparer les 6 des 9 et que nous projetons les données d'apprentissage sur leur axe principal, chaque chiffre initialement représenté par un vecteur de 25 caractéristiques (rétine 5×5) est alors transformé en une unique caractéristique dont les distributions sont présentées figure 8.

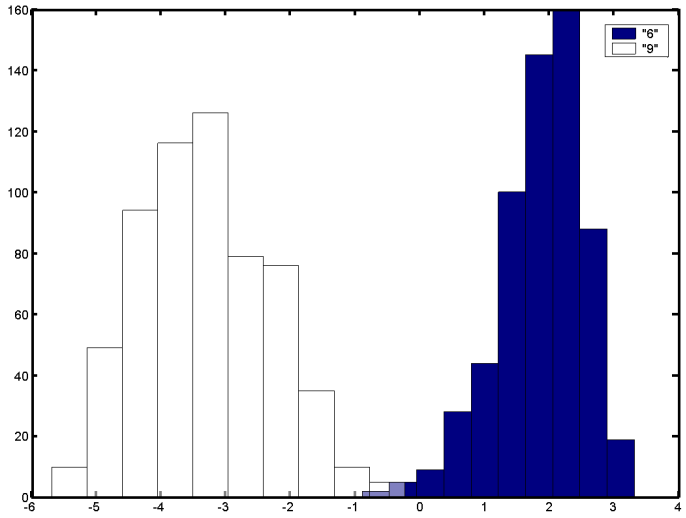


Figure 8 : Distribution des projections sur l'axe principal

Bien entendu, pour résoudre votre problème à 10 classes, il sera nécessaire d'utiliser plus qu'une seule composante principale. En effet, comme nous pouvons le constater en figure 9, la première composante exprime moins de 20% de la variabilité totale des données, alors que les 15 premières composantes en expriment environ 90%. Notons que ces résultats ont été obtenus en utilisant le script `calcul_acp.m` fourni en annexe.

Ensuite, vous pourrez utiliser le script `projection_acp.m` pour réduire la dimension du vecteur de caractéristiques en projetant les données sur les axes principaux.

Par ailleurs, au-delà des aspects de complexité de calcul et de capacité de stockage, la réduction de la dimensionnalité d'un problème peut aussi parfois permettre d'améliorer les performances de classification. Ainsi, le mathématicien Bellman, père de la programmation dynamique, a introduit l'expression « malédiction de la dimensionnalité » (curse of dimensionality), pour signifier que représenter les formes par des vecteurs de « grande » dimension est source de problème lorsque le nombre d'exemples d'apprentissage est limité.

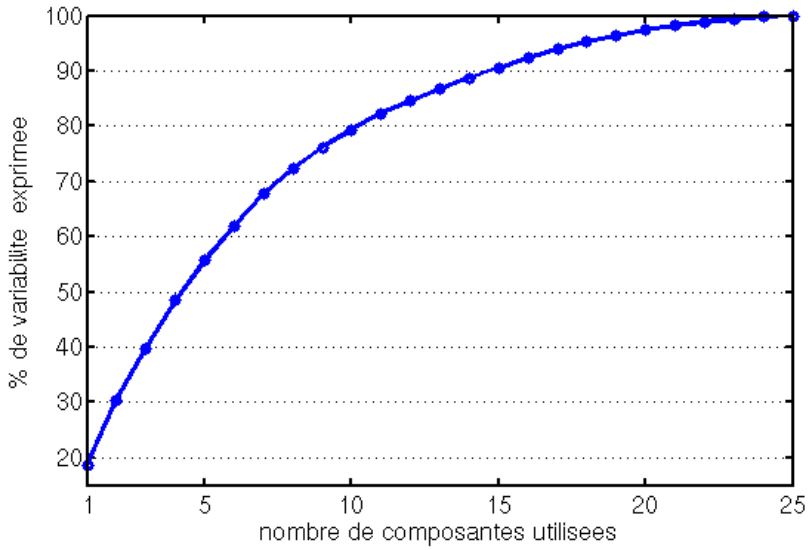


Figure 9: Variabilité exprimée en fonction du nombre de composantes

Travail à faire

I. Prétraitement et préparation de données : extraction de caractéristiques

1. Coder la fonction *retine.m* qui est utilisée dans le script *make_database.m* pour générer les caractéristiques et former les bases de données.
2. Générer diverses bases de données en faisant varier la taille de la rétine; puis étudier l'impact de la taille de la rétine sur le taux de recouvrement (ce taux permet de mesurer la capacité de discrimination des caractéristiques extraites).
    a) Utiliser les différentes métriques de similarité vues au Lab. pour calculer le taux de recouvrement. Discuter la sensibilité de taux de recouvrement aux différentes métriques de similarité.
    b) En fixant la taille de la rétine à son optimal (en utilisant par exemple la distance Euclidienne pour le calcul de recouvrement), extraire les caractéristiques de la base d'apprentissage.
    c) Calculer la divergence entre les classes deux à deux. Afficher le tout dans une matrice symétrique carrée. Calculer ensuite la divergence moyenne globale.

II. Réduction de la dimension

II.1. Par projection (ou transformation)

a. À l'aide de l'ACP, réduire la dimension de l'espace des caractéristiques. Il est demandé aussi de déterminer le nombre optimal de composantes principales.
b. Enfin, une fois les paramètres correspondant à la méthode Rétinal et aux nombres de composantes principales sont fixés, il vous faudra extraire les caractéristiques des images de la base de test. Attention, alors, il ne faut pas recalculer de nouveaux vecteurs propres mais utiliser ceux qui ont été déterminés à partir des données d'apprentissage.

II.2. Par sélection de caractéristiques

a. En utilisant le critère de divergence de ***Fisher***, sélectionner les ***n*** première caractéristiques les plus pertinentes. Le paramètre ***n*** peut être choisi par plusieurs stratégies. Pour ce laboratoire on choisit ***n*** manuellement.
b. Enfin, une fois les paramètres correspondant à la méthode Rétinal et aux nombres de caractéristiques sélectionnées sont fixés, il vous faudra sélectionner les caractéristiques les plus pertinentes des images de la base de test.

ANNEXES

make_database.m

```
clear all
path = 'Q:\SYS821\Publique\Database\Learning\';

nb_ex_class = 600;
database = zeros(nb_ex_class*10,25);
labels = zeros(1,nb_ex_class*10);

for i = 1:10
    tic
    fid = fopen([path 'listing_' num2str(i-1) '.txt'],'r');
    for j = 1:nb_ex_class
        tline = fgetl(fid);
        image = imread([path num2str(i-1) '\' tline]);
        database(nb_ex_class*(i-1)+j,:) = retine(image,5,5);
      % database(nb_ex_class*(i-1)+j,:)= profil(image,5,5);
        labels(nb_ex_class*(i-1)+j) = i;
    end
    fclose(fid);
    toc
end

save retine_5x5_learning database labels
% save profil_5x5_learning database labels
```

retine.m

```
function output = retine(image,nb_l,nb_c)
% output = retine(image,nb_l,nb_c)

output = ones(1,nb_l*nb_c);
...
```

profil.m

```
function output = profil (image,nb_l,nb_c)
%% output = profil (image,nb_l,nb_c)

output = ones(1,2*( nb_l+nb_c));

%% on redimensionne l'image dans une matrice nb_l x nb_c

%% on compte le nombre de pixels horizontalement (de la gauche vers la droite et dans le sens contraire)

%% on compte le nombre de pixels verticalement (du bas vers le haut et dans le sens contraire)

%% on concatène les résultats dans output
```

calcul_acp.m

```
clear all

%%% on charge les données %%%
load retine_5x5_learning
[nb_ex,nb_feat] = size(database);

%%% on calcule la moyenne et la matrice de covariance %%%
moyenne  =  mean(database);
S = cov(database);
%%% on calcule les vecteurs propres et les valeurs propres %%%
[vec_p, L] = eig(S);

%%% on trie les vecteurs propres selon les valeurs propres %%%
[val_p, ind] = sort(diag(L));
val_p = flipud(val_p);
ind = flipud(ind);
vec_p = vec_p(:,ind);

%%% on sauvegarde les vecteurs propres et la moyenne %%%
save acp_retine_5x5 vec_p moyenne
%%% on trace la variabilité exprimée en fonction du nombre de composantes %%%
plot(cumsum(val_p)/sum(val_p)*100,'ro-')
```

Projection_acp.m

```
%%% on charge les données %%%
load retine_5x5_learning
[nb_ex,nb_feat] = size(database);
%%% on charge les vecteurs propres et la moyenne %%%
```

```
load acp_retine_5x5
%%% on projecte sur les n axes principaux %%%
n = 15;
database = (database-ones(nb_ex,1)*moyenne)*vec_p(:,1:n);

%%% on sauvegarde les nouvelles données %%%
save retine_5x5_acp_15_learning database labels
make_TestingSet.m

clear all
path = 'Q:\SYS821\Publique\Database\Test\';

nb_ex_class = 100;
data_test = zeros(nb_ex_class*10,25);
labels = zeros(1,nb_ex_class*10);

for i = 1:10
   tic
   fid = fopen([path 'listing_' num2str(i-1) '.txt'],'r');
   for j = 1:nb_ex_class
      tline = fgetl(fid);
      image = imread([path num2str(i-1) '\' tline]);
      data_test (nb_ex_class*(i-1)+j,:) = retine(image,5,5);
          % data_test(nb_ex_class*(i-1)+j,:)= profil(image,5,5);
      labels(nb_ex_class*(i-1)+j) = i;
   end
   fclose(fid);
   toc
end

%%% on charge les vecteurs propres et la moyenne calcules avec la base d'apprentissage %%%
```

```
load acp_retine_5x5
%%% on projecte sur les n axes principaux %%%
n = 15;
data_test = (data_test-ones(nb_ex,1)*moyenne)*vec_p(:,1:n);

%%% on sauvegarde les nouvelles données %%%
save retine_5x5_acp_15_test data_test labels
```