Current subjects in computer science

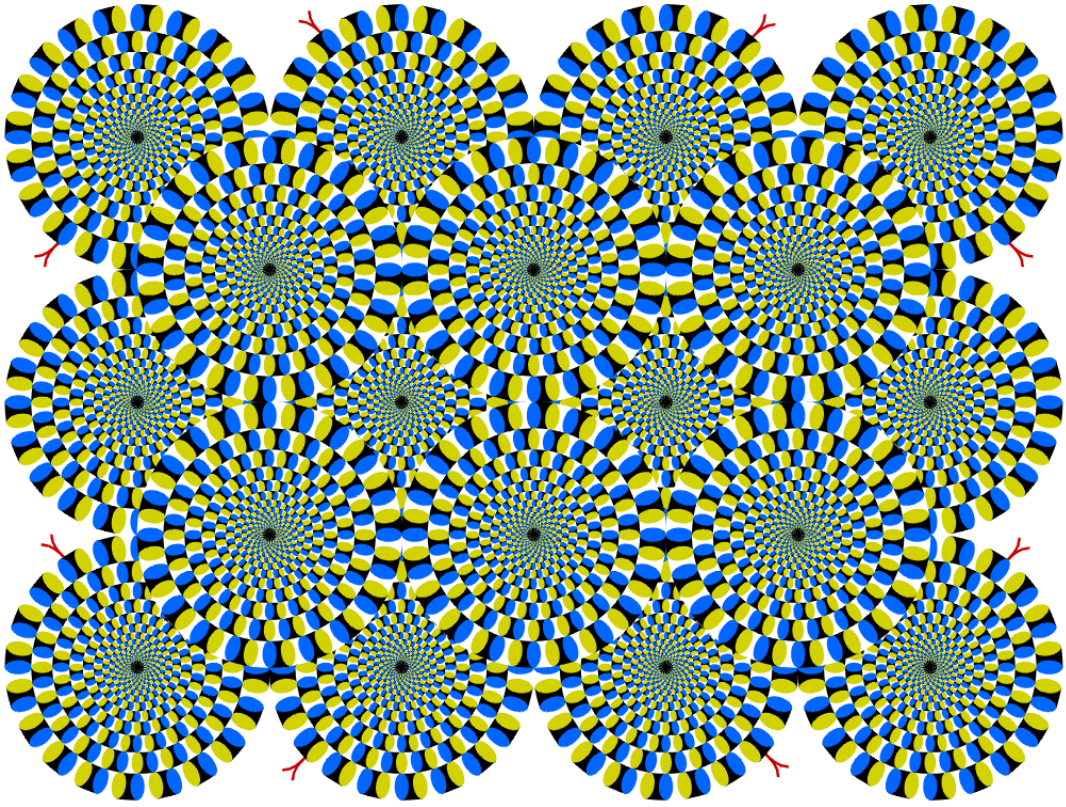
PATTERN RECOGNITION

PRACTICE 1:

- Studying PCA with a bidimensional set in Matlab.

- Studying PCA with images in Matlab.

- Face recognition sytem based on PCA.



1. Studying PCA with a bidimensional set in Matlab.

Run the matlab script ***PCA_example.m*.** This script computes the PCA directions of the bidimensional dataset formed by the data matrix X.

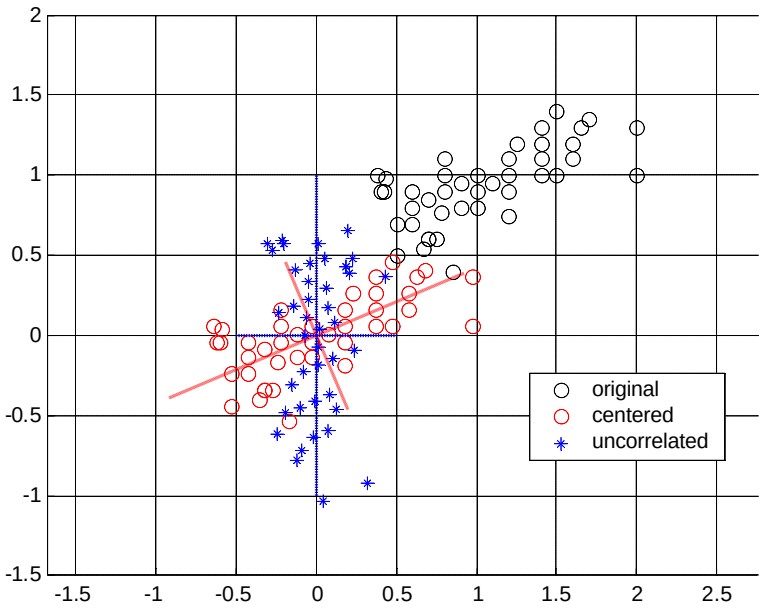


Figure 1. Output plot from ***PCA_example.m*.**

> ***Exercise 1*:** Using Matlab, answer these questions:
> - Which is the value of the mean vector of the original dataset?
> - Which is the value of the covariance matrix?
> - Which are the values of the eigenvectors and eigenvalues?

Exercise 2*:** Modify the ***PCA_example.m script in order to include the following data (X) and show the results graphically (as the previous *Figure1*):

X
(2,1)
(2,3)
(4,3)
(5,3)
(5,6)
(6,7)
(9,5)
(9,9)

Answer these questions:
- Which is the value of the mean vector of the original dataset?
- Which is the value of the covariance matrix?
- Which are the values of the eigenvectors and eigenvalues?

2. PCA examples with images

The *Figure 2* illustrates the performance of the PCA transformation with images. Basically, this figure describes the processes: eigenspace generation and reconstruction.

On the top, the first row shows the color images of 9 well-known motorcycle riders[1], these are the input images used to generate the PCA subspace. The input images are the original data matrix X of size M x N, where N=9 is the number of images, and M is the dimension of each image (M=273 x 254 x 3 = 208026).

On the second row, the first 8 eigenvectors of the PCA transform of X are shown. And the average image of the set is shown on the right.

The rest of the rows show the reconstructions of each image using subsets of eigenvectors: using 1, 2, 3,..8 vectors. For each rider, the image on the left is made using just one projection coefficient and 1 vector, and the image on the right is made using the whole set of vectors and so, it is equal to the original one.

At the end of each image set the reconstruction error is shown.
The *figure 2* has been drawn using the matlab function ***riders.m.*** (inside the **riders** folder)

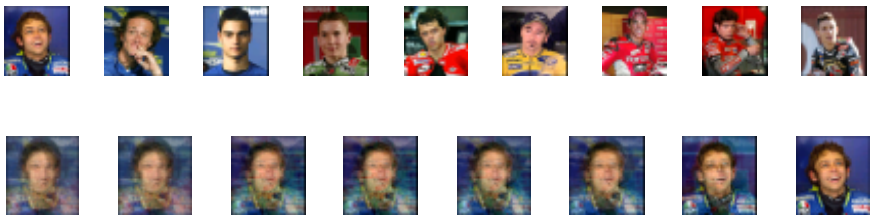
[1] *From left to right: Valentino Rossi, Sete Gibernau, Dani Pedrosa, Jorge Lorenzo, Loris Capirossi, Max Biaggi, Toni Elías, Roberto Rolfo y Manuel Poggiali.*

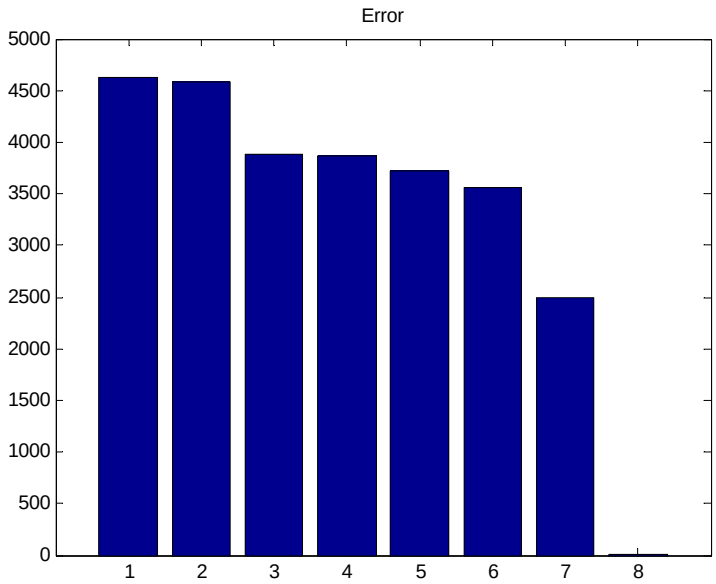
Figure 3. PCA example with images (famous motorcycle rider faces).

You must study carefully the matlab code of the function ***riders.m*** and run it. Try with

```
>> [im_reco, e, V, M, D, energy]=riders(1);
```

and you will obtain the following graphics:





In ***riders.m*** code, the PCA transformation is made by the function ***pc_evectors.m,*** which basically makes the same operations of ***PCA_example.m,*** but using patterns (images) of very high dimensionality (the color images of the riders).

- ***Exercise 3*****:** Modify ***riders.m*** in order to include the German rider Alex Hofmann (*hofmann.jpg*). You may eliminate any of the other riders.

3. Face Recognition System based on PCA.

There are a lot of face databases on the internet which the researchers use to test their new face recognition algorithms. You may find some of these databases in:

http://www-2.cs.cmu.edu/afs/cs/project/cil/ftp/html/v-images.html

In *Figures 4* and *5*, you may see some faces belonging to ORL face database. Basically, a face database is a collection of frontal face pictures of people.

In this section, we will use ORL face database (http://www.uk.research.att.com/facedatabase.html) to show how to make a face recognition system based on PCA. There are 10 images per person. And in our code we will use just 8 persons to show how the face system works.

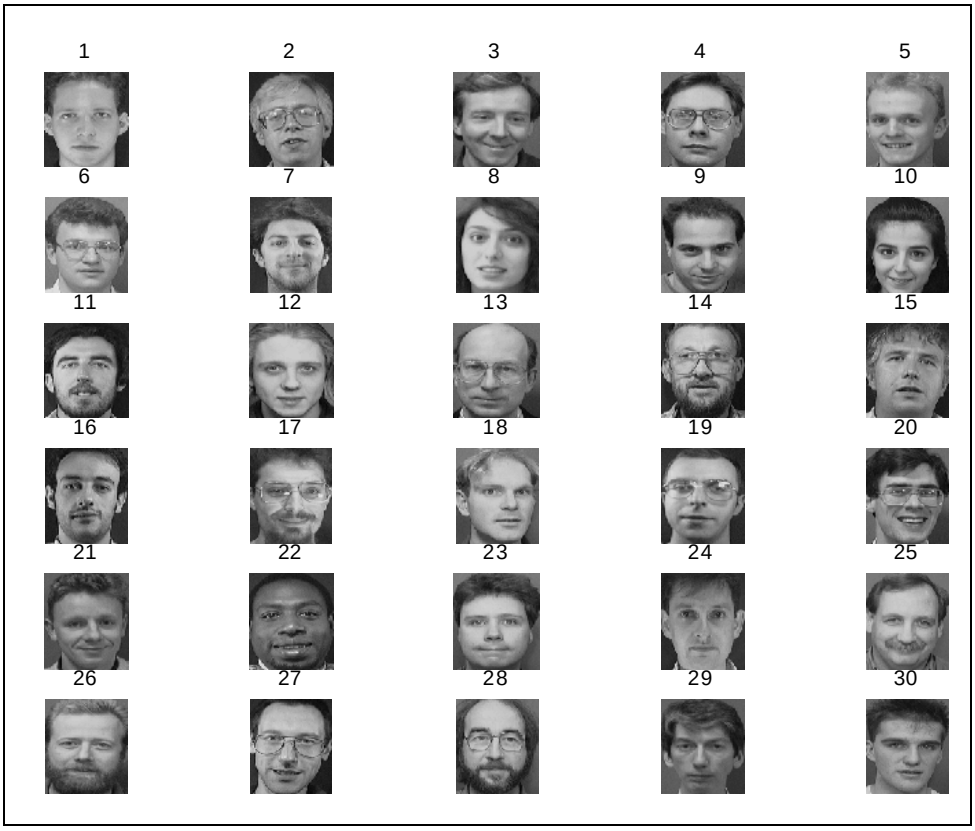


Figure 4. 30 different members of the ORL face database .

Figure 5. 10 images from the same person.

The matlab script ***ORL_faces.m*** ***(***inside the **faces** folder***)*** generates the PCA subspace using one image example per person and projects the rest of the images. Open and read the script carefully.

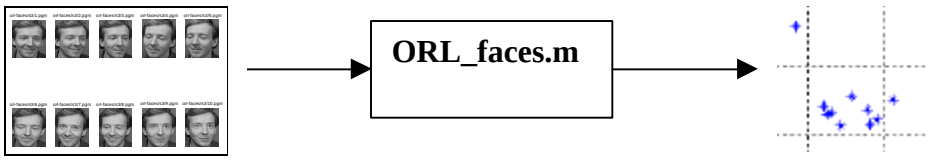


Figure 6.

Run:

```
>> [p1,p2,p3,p4,p5,p6,p7,p8]=ORL_faces(6,1);
```

The matlab script ***ORL_faces.m*** produces the PCA projections of the ORL faces. Each column in the matrices (***p1, p2***, ...) corresponds to a compressed face in a lower dimension space.

The 2D and 3D output graphics of ***ORL_faces.m*** give some idea of how the faces are separated. (see *Figures 7 and 8.*)

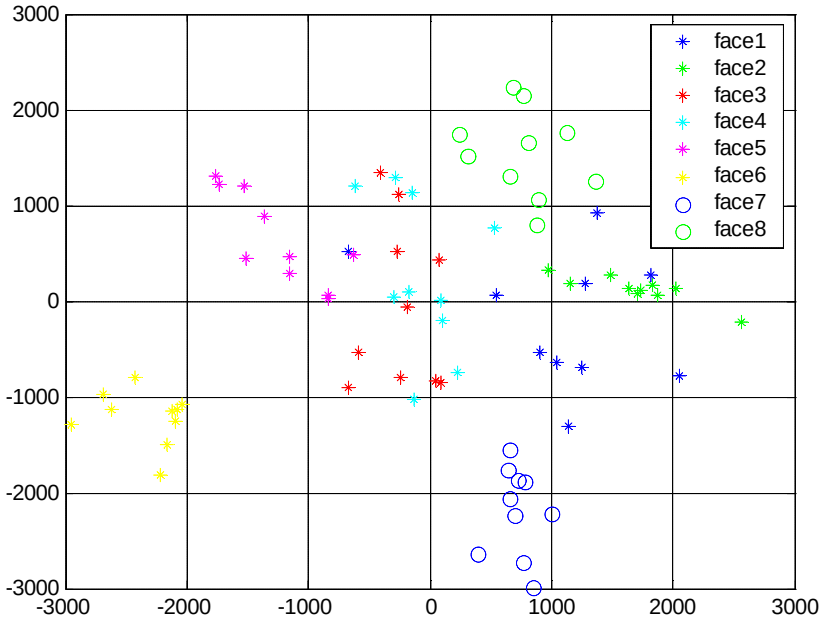
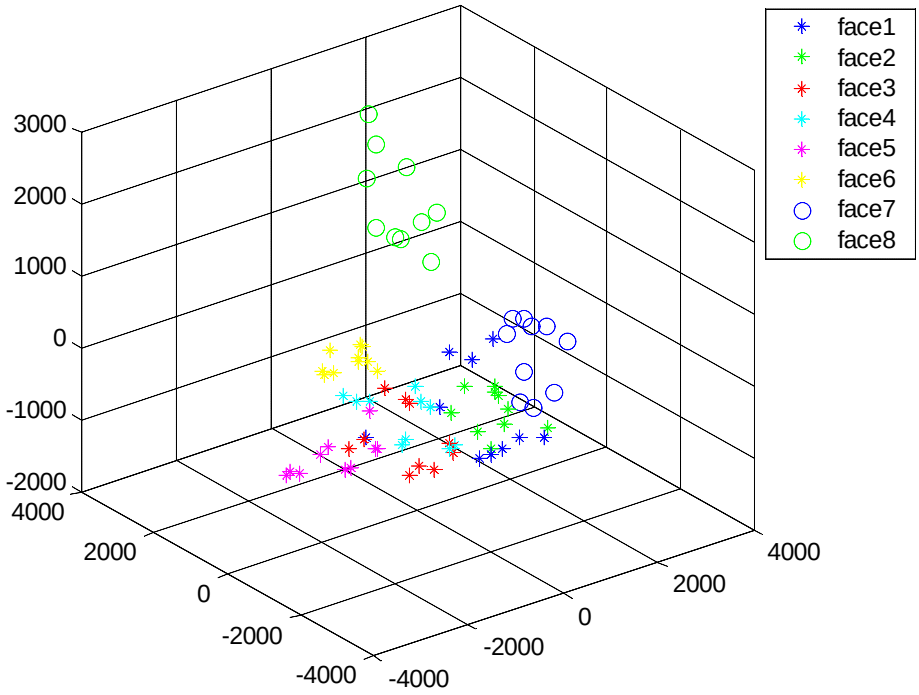


Figure 7. Bidimensional projections of the faces from ORL



Figures 8. 3D Projections of the faces from ORL

To test the performance of the recognition system, we will separate the face images in 2 sets:

- training
- test

and we will use the function ***classifier.m.***

```
function class=classifier(p1,p2,p3,p4,p5,p6,p7,p8,new_point,N_train,display)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
classifier --> compute the euclidean distance of the new_point to the class means
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p1,p2,p3,p4,p5,p6,p7,p8 --> the projections of the faces on the eigenspace
new_point --> new projection, we want to classify it
N_train --> number of training images
display ==1 show the graphics display==0 doesnt show the graphics
class -->  the result of the new_point classification
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The first eight parameters, **p1** to **p8**, correspond to the projections of the faces of eight different persons on the eigenspace. These are the training examples for each of the eight classes of the problem. The next parameter, **new_point**, corresponds to the new face image that has to be classified (actually, **new_point** contains the projection of such new face image on the eigenspace). The function **classifier.m** computes the distance from the new point to each of the classes, and selects the nearest class (i.e. the person the new image is supposed to belong to).

The **N_train** parameter is used to specify the number of training examples to be used in the experiment (i.e. the number of points used to compute the mean value of each class, which will be used for measuring distances to the new face). In general, the higher the number of training examples, the better the behaviour of the system.

N-train has to be lower than 10, as the number of available examples per class (images per person) is 10.

Finally, the display parameter is used to show different plots which help understanding the class distribution of the problem.

If you run:

```
>> classifier(p1,p2,p3,p4,p5,p6,p7,p8,p1(:,10),5,1)
```

you will obtain:

```
ans =

     1
```

That means the new point (p1(:,10)) belongs to class 1, so it´s a face from person number 1 of ORL.

The function ***classifier.m.*** also provides some ouput graphics. One of this graphics is shown in *Figure 9.*

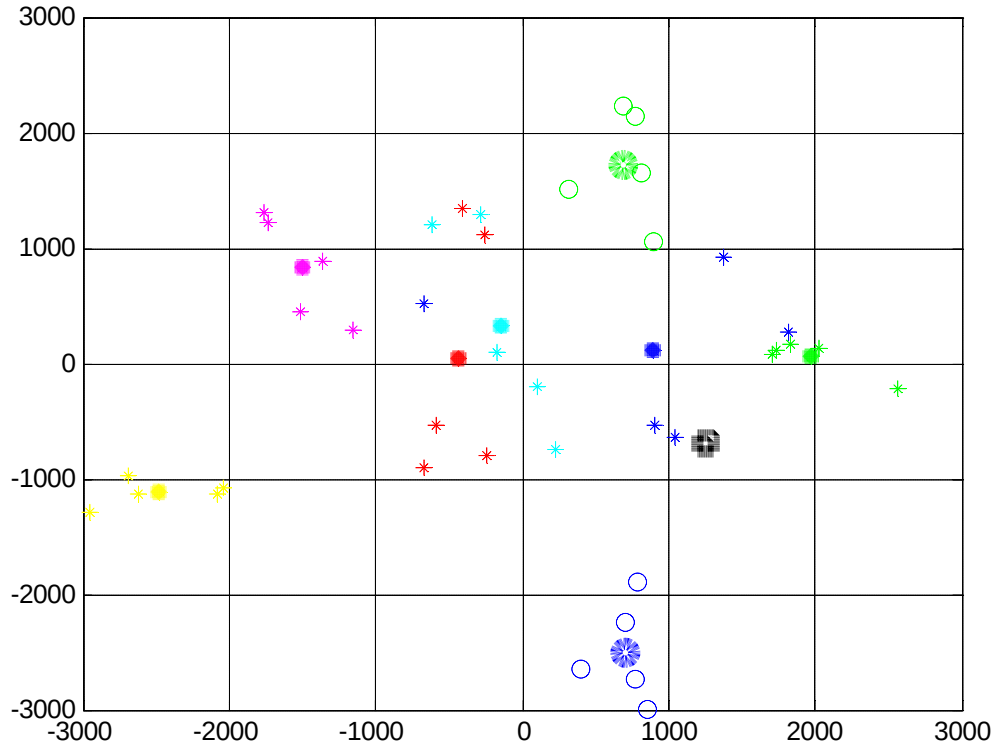


Figure 9. Output graphic from ***classifier.m***

Finally, it is possible to measure the expected classification accuracy of the face recognition system. In order to obtain such measure, you have to run the function ***classifier.m*** for many different images and compute how many correct classifications are obtained. This process is carried out by ***test_faces.m***

```
Nv=3; %dimension of the projections
N_train=5;%number of training images
N_test=5; %number of test images

[p1,p2,p3,p4,p5,p6,p7,p8]=ORL_faces(Nv,0);

results=zeros(8,N_test);
for j=1:8  %8 members
        for i=(10-N_test+1):10 %N_test faces to test

comando=sprintf('results(%d,%d)=classifier(p1,p2,p3,p4,p5,p6,p7,p8,p%d(:,%d),%d,0
);',j,(i-N_train),j,i,N_train);
       eval(comando);
       end
       end
results
```

If you run the previous program, you will obtain:

results =

1	**1**	**2**	**1**	**1**
1	**2**	**2**	**1**	**2**
4	**4**	**4**	**3**	**3**
1	**4**	**4**	**3**	**3**
5	**5**	**3**	**3**	**4**
6	**6**	**6**	**6**	**6**
7	**7**	**7**	**7**	**7**
8	**8**	**8**	**8**	**8**

That result has to be interpreted as follows:
- The first row of the matrix corresponds to the class predicted for each of the **5** training images of **person #1**. One of these images has been classified as belonging to **person #2**, so there has been a classification error. The remaining images have been correctly classified.
- The remaining rows have to be interpreted in the same way.
- Having a look at the whole matrix, it should be clear that there have been **12** classification errors out of **40** experiments.

From the previous result, it is possible to compute the **confusion matrix**. Such matrix represents the classification results in a slightly different way: each row of the confusion matrix is associated to an actual class and each colum of the matrix is associated to a predicted class. Ideally, the confusion matrix should be diagonal (when there are no classification errors).

Class	1	2	3	4	5	6	7	8	Total	Success	Error
1	4	1	0	0	0	0	0	0	5	80%	20%
2	2	3	0	0	0	0	0	0	5	60%	40%
3	0	0	2	3	0	0	0	0	5	40%	60%
4	0	1	2	2	0	0	0	0	5	40%	60%
5	0	0	2	1	2	0	0	0	5	40%	60%
6	0	0	0	0	0	5	0	0	5	100%	0%
7	0	0	0	0	0	0	5	0	5	100%	0%
8	0	0	0	0	0	0	0	5	5	100%	0%
Total	6	4	6	5	2	5	5	5	20	**70%**	**30%**

- ***Exercise 4*:** Run ***test_faces.m*** with these parameters:
  - N_train=5, N_test=5, Nv=6
  - N_train=2, N_test=8, Nv=6

and in each case, compute the confusion matrix. You may do it manually or using a matlab function. It´s up to you!