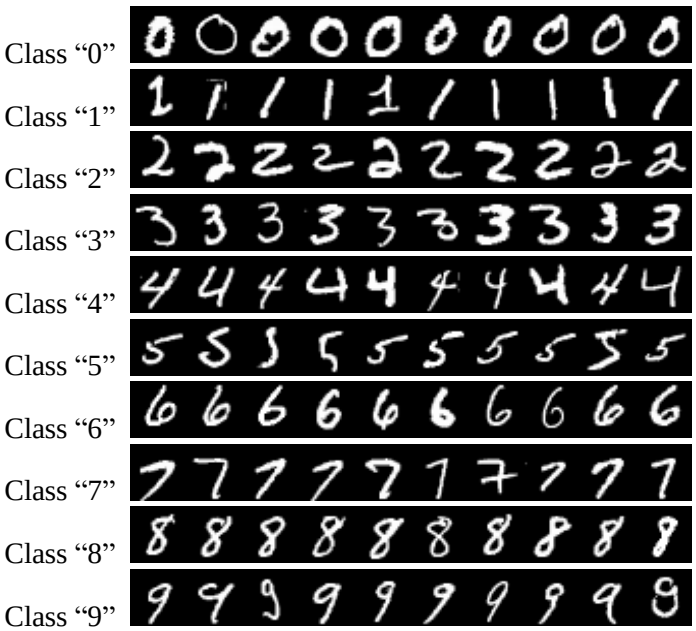
Dpto. Ingeniería de Sistemas y Automática
Universidad de Málaga

COMPUTER VISION

EXERCISE: Pattern recognition

Conceptos: Binary classifier

Design a binary classifier to identify numbers with only one digit. For the estimation of the probability distributions utilize the training images (28x28 pixels) that can be found in the file **train_data**. That file contains 500 images of each digit (class).



For doing that, you have to accomplish the following steps:

Design, Training

1. For each class **i**, i=1..9, read the training images and transform them into column vectors. Store them into a 3D matrix with size (28x28,500,10). *Note: use the Matlab command **reshape**.*

```
[…]
for type = 1:N_images_types
    for i_image = 1:N_images
        im_file_name=strcat(path_train,int2str(type-
1),'_',int2str(i_image),'.png');
        im = imread(im_file_name);
        images(:,:,i_image,type)=im;
    end
end
```

2. For each class **i**, compute the probability distributions p^i_f (f=1…28x28) with the training data obtained in the previous point. Store the probabilities in a matrix (28x28,10).

3. Finally, compute the vector of weights using such probability distributions. You have to compute w_j, j=1..n, with **n** the number of features (28x28 in this case), and also w_{n+1} (check the subject slides). *Note: Take care with ln(0)!! Use values close to 1 or 0, but not them!*

```
for type = 1:N_images_types
    x=images(:,:,:,type)~=0;
    p_f(:,:,type)=Change_prob(sum(x,3)/N_images);%Change 0 to 0.001
and 1 to 0.9999
    [w_f(:,:,type),
w_n1(type)]=Parts_disc(p_f(:,:,type),1/N_images_types);
end
function [wf wn] = Parts_disc(pf,ck)
    wf=log(pf./(1-pf));
    pf=reshape(pf,[1 size(pf,1)*size(pf,2)]);
    wn=log(ck)+sum(log(1-pf));
end
```

Testing

4. Once the classifier is trained, read the images in the **test_data** file and classify them.

```
%Testing
path_train = 'test\timage';
for i=1:length(digits)
    im_file_name = strcat(path_train,int2str(i),'.png');
    x = imread(im_file_name);
    for j=1:N_images_types
        dx(j)=discriminants(w_f(:,:,j),w_n1(j),x);
    end
    [~,pos] = max(dx);
    im=extractAfter(im_file_name,5);
    %fprintf('The image %s is clasified as %d\n',im,pos-1);
    class(i)=pos-1;
end
function dx = discriminants(wf,wn,x)
    wf=reshape(wf,[1 size(wf,1)*size(wf,2)]);
    x=(reshape(x,[1 size(x,1)*size(x,2)]))~=0;
    p = sum(wf.*x);
    dx=wn+p;
end
```

5. Give the final classifier success (100*number of digits correctly classified/total number of digits). For that you need to load the **digitos.mat** file, which contains the digit appearing in each test image.

```
digits = load('test\digitos.mat');
digits = digits.digitos;
prob=Calculate_prob(class,digits);
fprintf('The percentage of successful classify is %d%%\n',prob);
function prob=Calculate_prob(class,digits)
    eq=class'==digits;
    prob=sum(eq)*100/length(eq);
end
```

This code when it is executed gives an 84 % of correct classification, which means that the binomial classifier is good to classify this type of data.

Testing with your own data

6. The idea here is to take a picture of a piece of paper with digits on it with your phone/webcam, and try to classify them. For that, implement a code that, given an image:
    a. Manually extracts a squared region including only a digit. The digit has to occupy most of the region. *Note: use the Matlab **ginput** command.*

```
[X,Y] = ginput(2);
im=im<200;
rec=[X(1) Y(1) abs(X(1)-X(2)) abs(Y(1)-Y(2))];
imth=imcrop(im,rec);
```

    b. Modifies the region size to obtain another image with 28x28 pixels. *Note: use the Matlab **imresize** command.*

```
imth = imresize(imth,[28 28]);
```

    c. Provides a column vector with the information in such region.
    d. Computes the values of the decision function of each class, and provides the belonging class of such digit (the one with the highest value).

```
for j=1:N_images_types
    dx(j)=discriminants(w_f(:,:,j),w_n1(j),imth);
end
[~,pos] = max(dx);
subtitle = sprintf('The chopped image is clasified as %d\n',pos-
1);
xlabel(subtitle);
function dx = discriminants(wf,wn,x)
    wf=reshape(wf,[1 size(wf,1)*size(wf,2)]);
    x=reshape(x,[1 size(x,1)*size(x,2)]);
    p = sum(wf.*x);
    dx=wn+p;
end
```