PRÁCTICA 2 - RECONOCIMIENTO AUTOMÁTICO DE DÍGITOS MANUSCRITOS

Data

Data loading

```
% Data parameters
imgFile = "train-images-idx3-ubyte";
labelFile = "train-labels-idx1-ubyte";
readDigits = 100;%10000;
offset = 0;

[imgs labels] = readMNIST(imgFile, labelFile, readDigits, offset);

% Setting random numbers seed
rng(0, 'twister');
```

Features vectors generation & data set construction

```
featuresVectorsOfImgs = [];

% Features vectors generation
for i = 1:readDigits
    fv = featuresVector(imgs(:, :, i));
    featuresVectorsOfImgs = [featuresVectorsOfImgs ; array2table(fv)];
end

featuresVectorsOfImgs.Labels = labels;
```

Split of set in training set (80%) and test set

```
% Trainin data: 80% ; Test data: 20%
p = cvpartition(readDigits, "HoldOut", 0.2);
i = p.test;

% Splitting data
trainingData = featuresVectorsOfImgs(~i, :);
testData = featuresVectorsOfImgs(i, :);
```

Testing ML Models

```
% Exported models: quadraticSVM
load quadraticSVM.mat

testModel(quadraticSVM, testData)
```

Functions

Features vector generation function

```
function features = featuresVector(img)
    img = imresize(img, [40, 40]);      %????
    img = im2bw(img, graythresh(img));

    features = [];

    % Concavities
    cf = concavitiesFeatures(img);
    features = [features cf];

    % Holes
    hf = holesFeatures(img);
    features = [features hf];

    % Convex Hull
    chf = convexHullFeatures(img);
    features = [features chf];
end
```

Feature extraction functions

Concavities

The extracted concavities features are (for each concavity): centroid_x, centroid_y, area, major_axis_lenght, minor_axis_lenght.

```
function cf = concavitiesFeatures(img)
    imgWithoutHoles = imfill(img, 'holes');

    ch = bwconvhull(imgWithoutHoles, 'objects');

    concavities = ch-imgWithoutHoles;

    [eti num] = bwlabel(concavities,4);

    Dades = regionprops(eti,'all');

    N = 5;
    if size(Dades, 1) > 1
        T = struct2table(Dades);
        sortedT = sortrows(T, 'Area', "descend");
        Dades = table2struct(sortedT);
        Dades = Dades(1:min(size(Dades, 1), N));
    end

    centroidsX = zeros(1, N);
    centroidsY = zeros(1, N);
    area = zeros(1, N);
    majorAxisLength = zeros(1, N);
    minorAxisLength = zeros(1, N);

    for i=1:size(Dades, 1)
        dada = Dades(i);
        centroidsX(1, i) = dada.Centroid(1);
        centroidsY(1, i) = dada.Centroid(2);
        area(1, i) = dada.Area;
        majorAxisLength(1, i) = dada.MajorAxisLength;
        minorAxisLength(1, i) = dada.MinorAxisLength;
    end

    cf = [centroidsX centroidsY area majorAxisLength minorAxisLength];
end
```

Holes

The extracted holes features are (for each hole): centroid_x, centroid_y, area, major_axis_lenght, minor_axis_lenght.

```
function hf = holesFeatures(img)
    imgWithoutHoles = imfill(img, 'holes');

    holes = imgWithoutHoles-img;

    [eti, num] = bwlabel(holes,4);

    Dades = regionprops(eti,'all');

    N = 2;
    if size(Dades, 1) > 1
        T = struct2table(Dades);
        sortedT = sortrows(T, 'Area', "descend");
        Dades = table2struct(sortedT);
        Dades = Dades(1:min(size(Dades, 1), N));
    end

    centroidsX = zeros(1, N);
    centroidsY = zeros(1, N);
    area = zeros(1, N);
    majorAxisLength = zeros(1, N);
    minorAxisLength = zeros(1, N);

    for i=1:size(Dades, 1)
        dada = Dades(i);
        centroidsX(1, i) = dada.Centroid(1);
        centroidsY(1, i) = dada.Centroid(2);
        area(1, i) = dada.Area;
        majorAxisLength(1, i) = dada.MajorAxisLength;
        minorAxisLength(1, i) = dada.MinorAxisLength;
    end

    hf = [centroidsX centroidsY area majorAxisLength minorAxisLength];
end
```

Convex hull

The extracted holes features are: centroid_x, centroid_y, area, major_axis_lenght, minor_axis_lenght.

```
function chf = convexHullFeatures(img)
    ch = bwconvhull(img, 'objects');

    [eti num] = bwlabel(ch,4);

    dada = regionprops(eti,'all');

    % si s'extrau més d'un objecte que agafi el més gran
    % ja que serà aquell que representi la convex hull
    if size(dada, 1) > 1
        T = struct2table(dada);
        sortedT = sortrows(T, 'Area', "descend");
        dada = table2struct(sortedT);
        dada = dada(1);
    end

    centroidsX = dada.Centroid(1);
    centroidsY = dada.Centroid(2);
    area = dada.Area;
    majorAxisLength = dada.MajorAxisLength;
    minorAxisLength = dada.MinorAxisLength;

    chf = [centroidsX centroidsY area majorAxisLength minorAxisLength];
end
```

Test model & metrics

For a given model and test set shows the confusion matrix, the accuracy, the recall and the precision of each class and general.

Also shows the F-Score and returns the prediction.

```
function predicciones = testModel(model, testSet)
    % Predicciones en test
    predicciones = model.predictFcn(testSet);

    % Extracción de información en test
    figure, confussionTest = confusionchart(testSet.Labels,predicciones), title('Testi

    % METRICS
    % Confusion matrix
    [confMat, order] = confusionmat(testSet.Labels,predicciones);

    % Accuracy
    accuracy = 0;
    for i =1:size(confMat,1)
        accuracy=confMat(i,i) + accuracy;
    end
    Accuracy = accuracy/size(predicciones,1)
    % Recall
    for i =1:size(confMat,1)
        recall(i)=confMat(i,i)/sum(confMat(:,i));
    end
    recall(isnan(recall))=[];

    recall
    Recall = sum(recall)/size(confMat,1)

    % Precision
    for i =1:size(confMat,1)
        precision(i)=confMat(i,i)/sum(confMat(i,:));
    end
    precision
    Precision = sum(precision)/size(confMat,1)

    % F-score
    F_score=2*Recall*Precision/(Precision+Recall)
end
```