Centre for Mathematical Sciences, Mathematics, LTH, September 2014

Image Analysis, Handin 3

In this assignment you will study machine learning and continue your work on your own Optical Character Recognition system. The data for the assignments is on the course homepage:
`http://www.ctr.maths.lu.se/course/newimagean/2014/`

The assignment is due on thursday of study week 5. Make sure you answer all questions and provide complete solutions to the exercises. You may hand in hand-written solutions and/or printouts in the slot marked "bildanalys" on the 3rd floor of the mathematics building and/or as a pdf by mail to fman20@maths.lth.se. Write your name and the assignment number in the subject line. For each exercise, we have tried to make it clear what should be included in the report. In addition, all the code should be submitted as m-files by mail. Make sure that your matlab scripts are well commented and can be executed directly (that is, without loading any data, setting parameters etc. Such things should be done in the script).

You will have time to work with the assignments during the computer laboratory sessions and the exercise sessions. These sessions are intended to provide an opportunity for asking questions to the lecturer on things you have problems with or just to work with the assignment. During the laboratory sessions you should work on the exercises marked "Computer Exercise". The rest of the exercises are intended to provide hints and prepare you for the computer exercises. You are expected to have solved these before you go to the lab sessions. The report should be written individually, however you are encouraged to work together (in the lab session you might have to work in pairs). Keep in mind that everyone is responsible for their own report and should be able to explain all the solutions.

1 Line fit

On the course homepage there is a link to the file `linjepunkter.mat`. Download this file and load it into matlab. Write two matlab routines. One for fitting a line to the points according to the least squares approach and one for minimising the squared distance to the line (the so called total least squares method, see lecture notes). What is the difference between the two methods. How big difference is there in the result?

For the report: Provide your code for line fitting in the report. Provide plots with the points and the fitted lines. Write a few sentences on the result. How did it go? How big difference is there between the two methods?

2 Your own classifier

To better understand machine learning it is useful to have coded your own classifier.

Most classifiers consists of two parts, a training part and a classification part. For the training part, one uses a training set consisting of example feature vectors x_i with corresponding ground truth y_i. The training set

$$T = \{(x_1, y_1), \ldots, (x_n, y_n)\}.$$

is sometimes represented as the matrices

$$X = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}$$

and

$$Y = \begin{pmatrix} y_1 & \cdots & y_n \end{pmatrix}.$$

Here the each feature vector x_i typically contains several features, i.e. x_i is a column vector, whereas y_i is a scalar that denotes the ground truth label.

Choose your favorite classification method (e.g. Bayes, Nearest Neighbour, or Support vector machine) and implement a routine for training

```
function [classification_data] = class_train(X,Y);
```

and a routine for classification

```
function y = classify(x,classification_data)
```

Now, some machine learning techniques are easier (nearest neighbor or bayesian classification assuming Gaussian distributions) and some are more difficult (Support Vector Machine) to code. Choose wisely!

Download dataset *FaceNonFace.mat*. It contains 200 feature vectors as columns in a matrix X and the corresponding ground truth in a vector Y. Extract from these a random partition for training. You can use `cvpartition` to create a random partition. By running

```
part = cvpartition(200,'HoldOut',0.20);
```

one can then get a random index set `part.training` of the training examples and `part.test` for testing.

Here `classification_data` is a variable that contains, whatever information you have extracted during training and is needed during testing.

How well does this work?

Try the method on the dataset *FaceNonFace.mat*. It contains a matrix X of size 361×200. Each column consists of 361 pixels from 200 small 19×19 pixel images. There are thus 200 such feature vectors. In the matrix Y he ground truth is given for the 200 examples. These are coded so that '1' corresponds to face and '-1' to non-face.

In the written solution to the segmentation problem, supply both code (e.g. matlab code) and a printout the results of using your algorithm, i.e. supply examples of input data (e.g. as images) and result after applying your classification algorithm (e.g. as a table). Also provide error rates for your classifiers both on training and testing data.

3 Use pre-coded machine learning techniques

There are many machine learning techniques. Most of them have a training part and a classifying/evaluating/testing part. Some of these take considerable effort to code. It is important to be able to re-use code. One of the difficulties here lies in understanding the inputs and outputs of such code. In this exercise you will practice this.

On the course homepage is a file `ocrfeaturestrain.mat` that contains a matrix X of size 13×570. Each column consists of 13 features calculated from one segmented character from another OCR dataset, where the font size varies. There are thus 570 such feature vectors. In the matrix Y he ground truth is given for the 570 examples. These are coded from 1 to 26, where 1 corresponds to 'A', 2 corresponds to 'B' and so forth.

From the dataset, select two classes, e.g. the letters 'O' and 'C'.

Try at least three different machine learning techniques. Select a random training subset, e.g. using mat labs function `cvpartition`. You can either use your own training and testing routines or use some of mat labs routines, e.g.

```
T = classregtree(X,Y,'method','classification')
svmStruct = svmtrain(X,Y);
knn = ClassificationKNN.fit(X,Y);
```

If you use these routines you have to transpose the matrices, since they assume that each example is a row in the matrix.

See `help cvpartition`, `help classregtree`, `help svmStruct` or `help ClassificationKNN.fit`.

In the written solution to the segmentation problem, supply both code (e.g. matlab code) and a printout the results of using your algorithm, i.e. supply some examples of input data (e.g. as images) and result after applying your classification algorithm (e.g. as a table). Also provide error rates for your classifiers both on training and testing data.

4 OCR system construction and system testing

Here we will continue the work on our OCR system. From the two previous assignments you have constructed

- `S = im2segment(Im)` - a segmentation algorithm that takes an image to a number of segments.
- `x = segment2features(S)` - an algorithms for calculating a feature vector x from a segment S.

Now using machine learning you will construct a classifier that takes a feature vector x and calculates a class y.

On the course homepage, there is a matlabfile `ocrsegments.mat`. It contains a number of segments in a cell array S and corresponding letter code y. Again these are coded from 1 to 26, where 1 corresponds to 'A', 2 corresponds to 'B' and so forth.

Use your own set of favorite features. Use your favorite machine learning method to train a classifier.

Use this classifier to code a complete OCR-system consisting of your three steps The system should take an image I as input and return a string of ascii characters s,

```
function s = ocr(I,classdata)
```

Download the program `benchmark.m` from the home-page and study the program. The purpose of the program is to benchmark your system and calculate the percentage of characters that are correctly found and classified. Such test routines are important, when improving different parts of the program. It is important to have automatic test routines to verify that new versions are better or at least as good as old versions.

Benchmark your system with the images in the folder `short1`,

```
% Supply the name of your ocr function
myocr = 'ocr';
% Supply the path to your data folder
datadir = '/usr/matematik/kalle/matlab/ocr/datasets/short1';
% Benchmark your ocr function
[hitrate,confmat,res]=benchmark(myocr,datadir,classdata);
% Display the result
disp(['Using method ' myocr ' on dataset ' datadir ...
' I obtained a hitrate of ' num2str(hitrate)]);
```

Try the code both on datasets 'short1', 'short2' and 'home2'.

Discussion: It is important to test your image analysis system on real data with ground truth. This makes it possible to check your system and verify that it works. However, if the system does not work, a high error rate can be quite uninformative. It is difficult to know if it is the segmentation part that fails, if it is the features that are not good enough or if it the classification algorithms that fails. Think about this and discuss (briefly) in the report what one could do to shed more light on a system that fails.

In the written solution to the segmentation problem, supply both code (e.g. matlab code) and a printout the results of using your algorithm, i.e. supply some examples of input data (e.g. as images) and result after applying your classification algorithm (e.g. as a table). Also provide error rates for your classifiers both on training and testing data.