Computer exercise 5
Image classification
Spatial statistics and image analysis, TMS016

1 Introduction

The purpose of this computer exercise is to give an introduction to how image classification can be performed in Matlab. When in doubt about how to use a specific function in Matlab, use `help` and `doc` to get more information.

2 Classification of handwritten digits

Throughout this exercise, we will use the first 1 000 digits from the training data in the MNIST database (see `http://yann.lecun.com/exdb/mnist/`). However, if the third part of your project assignment is about image classification, you can of course use the data from your project instead.

The MNIST data is contained in the file `mnist_data.mat`, where the variable `x` contains the images and the variable `z` the corresponding labels.

- Plot some of the images, the ith image is given by `x(:,:,i)`.
- Extract d features from the images and store them in an $1000 \times d$ matrix `m`. You can for example compute moments using the functions `image_moment`, central moments using `central_moment`, or all Hu-moments using `hu_moments`.
- Train a linear discriminant classifier using

  ```
  t = templateDiscriminant('DiscrimType','Linear')
  C = fitcecoc(m,z,'Learners',t);
  ```

- Plot the confusion matrix for the resubstitution errors using

  ```
  plotconfusion(categorical(z),categorical(resubPredict(C)))
  ```

  Note that in certain versions of Matlab (such as 2017a), the above command will not work. Then instead try

  ```
  f = @(X)double(bsxfun(@eq, X(:), 0:max(X)))';
  plotconfusion(f(z),f(resubPredict(C)));
  ```

- Compute predictions for a k-fold crossvalidation using

  ```
  Ccv = crossval(C,'kfold',k);
  zhat = kfoldPredict(Ccv);
  ```

  and plot the corresponding confusion matrix.

- You have now performed the basic steps for training and evaluating a classifier. Note the error rates for your classifier and try to improve them by either changing the features you use or by changing the classifier. To change classifier, simply change how `t` above is defined. See the documentation for `fitcecoc` for a list of implemented methods. For example, to use QDA use

```
t = templateDiscriminant('DiscrimType','Quadratic');
```

or to use a SVM with a Gaussian kernel, use

```
t = templateSVM('KernelFunction','gaussian');
```

Note that the SVM results can be sensitive to scaling of the features. If you get bad results, try

```
t = templateSVM('Standardize',1,'KernelFunction','gaussian');
```