Image Analysis
Exercise 07 - Pixel Classification

2019

Introduction

In this exercise we are going to implement pixelwise classification of different tissue types found in abdominal CT images. We also start making a program that can classify structures in natural images including road signs.

As usual create a Matlab m-file with your exercise.

Get the data from `courses.compute.dtu.dk/02502`.

Medical Image Segmentation

Training

The image we are going to work with is a DICOM image of a CT angiography examination of the liver. The raw DICOM image, where the pixels are stored as 16-bit values is `CTangio2.dcm`. A histogram stretched copy is called `CTAngio2Scaled.png`. The raw DICOM image should be used when pixel values are extracted, while the stretched image can be used for annotation (drawing regions).

Start by reading the images:

```
ct2 = dicomread('CTangio2.dcm');
I2 = imread('CTAngio2Scaled.png');
```

The goal is to label the different organs and structures seen in the image. Optimally, we would like to be able to recognise (this is our *classes*):

- Background
- Fat
- Liver
- Kidney

- Spleen
- Trabecular bone
- Hard bone

It is probably not possible to separate all the above tissue types. This is because their Hounsfield units are very similar. Hounsfield units are used in computed tomography to characterise the x-ray absorption of different tissues. A CT scan is normally calibrated so a pixel with Hounsfield unit 0 has an absorbance equal to water and a pixel with Hounsfield unit -1000 has absorbance equal to air.

Start by looking at the image and see if you can identify the different structures and organs. Use for example WikiPedia to find a guide to the internal organs.

```
imshow(I2);
```

The first thing we want to do, is to select a *training set* for our algorithm. We do that by selecting representative regions for each class.

Exercise 1 *Use* `roipoly` *to select representative regions of each of the seven classes. For example, to select a region for the liver and extract the raw 16-bit values, you can do like:*

```
LiverROI = roipoly(I2);
imwrite(LiverROI, 'LiverROI.png');
LiverVals = double(ct2(LiverROI));
```

Note that the background pixel values should be taken just above the stomach.

When you are done outlining an organ, you tell `roipoly` *that you are done by either double clicking the inside of the shape you created or right-clicking and selecting "Create Mask".*

In order to avoid having to annotate every time you run your script, you can later replace[1] *the above lines with:*

```
LiverROI = imread('LiverROI.png');
LiverVals = double(ct2(LiverROI));
```

You should now have seven arrays with values from the classes (`LiverVals`, `FatVals`, `KidneyVals`, ...).

Exercise 2 *Plot histograms of your training values using* `hist`*. Also calculate and print basic statistics of your class pixel values. You can for example use:*

[1] Comment out the lines, do not delete them entirely.

```
sprintf('Background mean %g std %g min %g max %d',...
mean(BackgroundVals), std(BackgroundVals), ...
 min(BackgroundVals), max(BackgroundVals))
```

Of the seven histograms, which would you say look like they belong to a Gaussian distribution?

The Matlab function `normpdf` represents a Gaussian probability density function. It can for example be used to plot Gaussian distributions with given mean and standard deviation.

Exercise 3 *To create a fitted Gaussian distribution of one of our histograms the following can be used*

```
figure;
xrange = -1200:0.1:1200; % Fit over the complete Hounsfield range
pdfFitLiver = normpdf(xrange, mean(LiverVals), std(LiverVals));
S = length(LiverVals); % A simple scale factor

hold on;
hist(LiverVals,xrange);
plot(xrange, pdfFitLiver * S,'r');
hold off;
xlim([-10, 100]);
```

We fit over the whole Hounsfield range. However, you can play around with `xlim` *to make a nice looking plot.*

The fitted Gaussians are good for inspecting class separation and overlap.

Exercise 4 *Create fitted Gaussians for all seven classes and plot them in the same plot. This can for example be done like:*

```
plot(xrange,pdfFitBackground, xrange, pdfFitTrabec, xrange, ...
    pdfFitLiver, xrange, pdfFitSpleen, xrange, pdfFitKidney, ...
    xrange, pdfFitFat, xrange, pdfFitBone);

  legend('Background','Trabeculae', 'Liver', 'Spleen',...
     'Kidney', 'Fat','Bone');
  xlim([-1200 1200]);
```

Which classes are well-separated and which overlap? Is it better to collapse some classes into one?

You should now select the classes that you believe can be identified.

Minimum Distance Classification

Exercise 5 *Use the principle of minimum distance classification to define the pixel value ranges of your classes. Write them in a table.*

Take a look at the supplied function `LabelImage` and try to understand what it does. `LabelImage` takes up to five range limiters as argument (`T1`, `T2`,...). If you do not use all of them, just set the last ones to the same value.

Exercise 6 *Use function* `LabelImage` *to create an image that is labelled (classified) according to your range definitions.*

```
ILabel = LabelImage(ct2, T1, T2, T3, T4, T5);
```

here `T1`, `T2` *are the range delimiters. For example could* `T1` *be the value that separates background from fat.*

Exercise 7 *To visualise the classes you can use the following:*

```
figure
imagesc(ILabel)
hcb=colorbar;
set(hcb,'YTick',[0,1,2,3,4,5]);
set(hcb,'YTickLabel',{'Class 0', 'Class 1','Class 2',...
'Class 3','Class 4','Class 5'});
```

You can change the tick labels to your chosen classes.

Does the classified image look like you expected?

Parametric Classification

When using the minimum distance classifier it is only the location of the top of the fitted Gaussian that is used. Now we also want to use the shape of the fitted Gaussian.

Exercise 8 *Look at the plot of all the fitted Gaussian distributions. Estimate the class boundaries by finding the intersection of the PDF's. You can for example use* `xlim` *to do this more precisely. Create a table with pixel value ranges for the parametric classifier.*

Exercise 9 *Create a new classified image using the ranges that you estimated for the parametric classifier. Compare it with the classification from the minimum distance classifier.*

DTU Sign Image Classification

The goal of this exercise is to be able to classify different objects in a color image. As an example we use the image `DTUSigns055.jpg`. From this image we can for example define the classes: `blue sign`, `red sign`, `white car`, `green leaves`, `yellow grass`. You can also define your own classes.

Exercise 10 *Use* `roipoly` *to select representative regions of each of the classes that you have chosen.*

If you for example have selected a `blue sign` region you can call it `BSROI`. To get the R, G, B values in that region you can for example do:

```
I = imread('DTUSigns055.jpg');
Ired   = I(:,:,1);
Igreen = I(:,:,2);
Iblue  = I(:,:,3);
redVals = double(Ired(BSROI));
greenVals = double(Igreen(BSROI));
blueVals = double(Iblue(BSROI));
```

It can be very useful to inspect the combined histogram of the R, G, B values. One way to do this is

```
figure;
totVals = [redVals greenVals blueVals];
nbins = 255;
hist(totVals,nbins);
h = findobj(gca,'Type','patch');
set(h(3),'FaceColor','r','EdgeColor','r','FaceAlpha',0.3,'EdgeAlpha',0.3);
set(h(2),'FaceColor','g','EdgeColor','g','FaceAlpha',0.3,'EdgeAlpha',0.3);
set(h(1),'FaceColor','b','EdgeColor','b','FaceAlpha',0.3,'EdgeAlpha',0.3);
xlim([0 255]);
```

Exercise 11 *Use the histograms to find class specific limits in RGB space to be able to segment you classes.*

You can use your limits to classify (segment) the entire image by for example doing

```
figure;
BlueSign = Ired > Rlo & Ired < Rhi  & Igreen > Glo &
 Igreen < Ghi & Iblue > Blo & Iblue < Bhi;
imshow(BlueSign);
```

where `Rlo`, `Rhi`, and so on are your limits for the RGB space.

Exercise 12 *Examine the resulting segmentations and discuss which of your classes that are easy to segment and which are hard?*

Exercise 13 *Take at least 5 photos from the DTU campus with at least some of your classes presents (signs, cars etc).*

Exercise 14 *Use the class limits that you learned before and see how well they work on your own images. This can for example be done like this:*

```
I2 = imread('MyOwnImageFromDTU.jpg');
Ired   = I2(:,:,1);
Igreen = I2(:,:,2);
Iblue  = I2(:,:,3);
figure;
BlueSign = Ired > Rlo & Ired < Rhi  & Igreen > Glo &
 Igreen < Ghi & Iblue > Blo & Iblue < Bhi;
imshow(BlueSign);
```

where `Rlo`, `Rhi`, *are the limits that you learnt using the training image.*