Matlab Exercises 2: Solutions

Luke Dickens

January 15th, 2015

Questions

You can find all the available files at my email address: `http://www.doc.ic.ac.uk/~lwd03`

1. Simply a list of the commands used for Q.10 on labsheet 1.
2. a) This will do it

```
load data
hist(X(1,:))
```

   b) Something like

```
function plothists(data)
    numplots = length(data(:,1));
    n = ceil(sqrt(numplots));
    for i = 1:numplots
        subplot(n,n,i);
        hist(data(i,:));
    end
end
```

   Data in `X` is all from gaussian distributions, but not all data from `Y` is.

3. Something like:

```
function [sequence] = mysequence(upperbound)
    sequence = [];
    for value=1:upperbound
        if test(value)
            sequence = [sequence value];
        end
    end
end
```

```
% the sub-function
function [ result ] = test(value)
    mag = floor(log10(value)) + 1;
    remainder = rem(value^2,10^mag);
    result = (remainder == value);
end
```

4. a) Pass each file name into the `importdata` function.

   b) This requires you to create folder `+lifelogging` and place the files `step_scores.m`, `activity_scores.m`, `mean_scores.m` and `summarise_data.m` into it.

   `step_scores.m` is straightforward.

   `activity_scores.m` will require using a vector like `[0,1/48,4/48, 20/48]`, then multiply every row by this. A good solution will use `bsxfun`, alternatively use `repmat`.

   For `mean_scores.m`, for each user id, find the rows that correspond, pull these out as a separate array, and then use `sum` on that.

   For `summarise_data.m`, again this is straightforward. Pass the relevant values to `step_scores` and `activity_scores` then pass the results to `mean_scores` before creating the struct array with the results.

   c) From the parent folder call the function `lifelogging.summarise_data`. The plotting is straightforward.

5. I used the following (see over), which operates on the clustermap (the second output) from the `segment_image` function. Although, there are other ways to do this.

```
function [boundaryimage,boundarymap] = get_boundary_image(clustermap)
% GET_BOUNDARY_IMAGE - takes clustermap of a segmented image and
% finds the boundaries
%
% Use:
% [ boundaryimage, boundarymap ] = get_boundary_image(clustermap)
% Takes:
% clustermap - H x W matrix of cluster ids
% Returns:
% boundaryimage - H x W x 3 image data of boundary image
% boundarymap - H x W matrix of boundary pixels

 % get image size
 mapsize = size(clustermap);
 % iterate over inner pixels to determine boundaries
 for i=2:mapsize(1)-1
  for j=2:mapsize(2)-1
   % the cluster id
   thisid = clustermap(i,j);
   % Does every neighbouring pixel have the same id?
   isboundary = all(all(clustermap(i-1:i+1,j-1:j+1) == thisid));
   % map this
   boundarymap(i,j) = isboundary;
   % choose pixel color
   boundaryimage(i,j,:) = isboundary*ones(3,1);
  end
 end
end
```