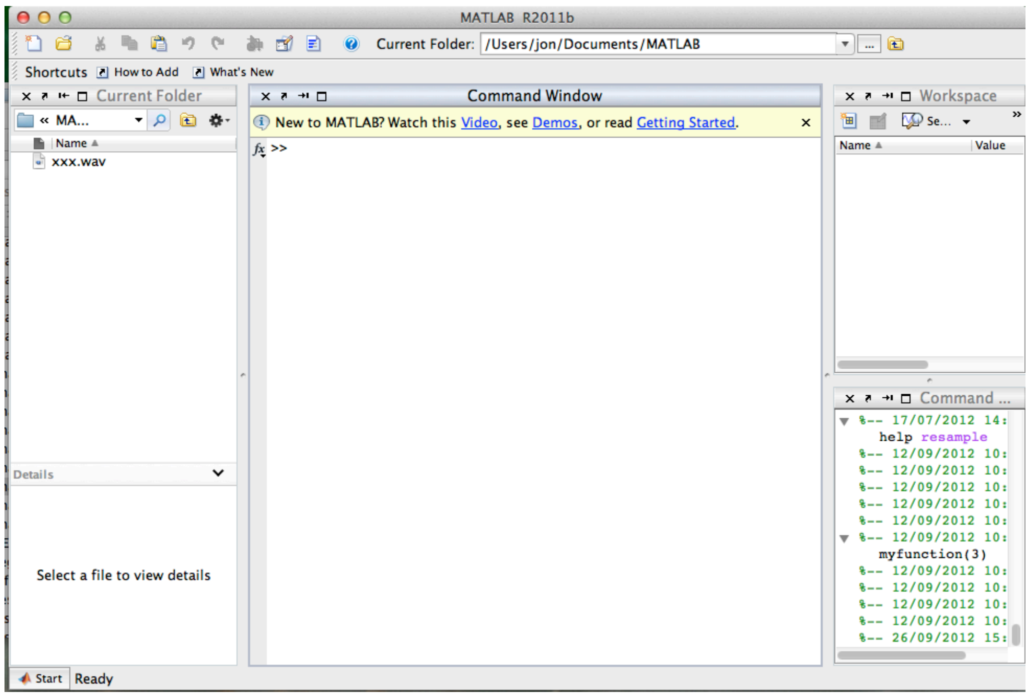
COM2004/3004 Lab Week 1
Getting Started with MATLAB – Diagnosing Liver Disease

Objectives

- **To provide an introduction to MATLAB.**
- **To introduce a classification task using real data.**
- **To learn how to use MATLAB to visualize 1- and 2-D distributions.**

1. Starting MATLAB

Boot your machine in Windows 7 and log in. Go to the Start menu and find MATLAB in the list of programs. Start MATLAB. You may have to wait some time but when MATLAB eventually starts you will see an interface looking something like the image below:



New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>>
```

```
%-- 17/07/2012 14:
   help resample
%-- 12/09/2012 10:
%-- 12/09/2012 10:
%-- 12/09/2012 10:
%-- 12/09/2012 10:
%-- 12/09/2012 10:
%-- 12/09/2012 10:
   myfunction(3)
%-- 12/09/2012 10:
%-- 12/09/2012 10:
%-- 12/09/2012 10:
%-- 12/09/2012 10:
%-- 26/09/2012 15:
```

The central pane is the *Command Window* into which you will be typing MATLAB commands.

2. What is MATLAB?

MATLAB is a program that was originally designed to simplify the coding of linear algebra routines (i.e. programs involving vectors and matrices). Over the years it has developed into something much bigger and is now the environment of choice for many people working in scientific computing, signal processing, machine learning etc. MATLAB has a shell-like interface that allows you to work interactively, but it also provides a simple yet powerful interpreted language and tools for building interfaces. It is also very convenient for plotting and visualizing complex data and has a wealth of 2-D and 3-D plotting commands.

3. Loading the lab data.

In this lab we are going to be experimenting with a set of medical data concerned with the diagnosis of liver disease. The data for this lab can be downloaded from my web page. Use a browser to visit the URL,

```
    http://staffwww.dcs.shef.ac.uk/people/J.Barker/COM2004.html
```

and click to download and save the data `liver_data.txt` somewhere on the Windows file system (e.g. either on the Desktop or in your student filestore).

MATLAB uses linux-like commands for navigating between folders and examining their contents. Experiment by typing the commands `ls`, `cd` and `pwd` at the MATLAB command prompt. Navigate around your file store. Note that the contents of the current directory are shown in the upper-left interface panel. For details on any MATLAB command you can type `help` followed by the command name, e.g. `help cd`.

Navigate to the directory where you saved the file `liver_data.txt`. Check it is there by using `ls` or by looking at the top-left interface panel and then load the data into MATLAB using,

```
    data = load('liver_data.txt');
```

Take care with the quotes and the semi-colon at the end.

This command will load all the rows of comma-separated numbers from the file liver_data.txt and store them all in a matrix variable called 'data'. (Note, the variable 'data' did not need to be declared or given a type, or allocated memory … MATLAB is very different from Java!) You will notice that the variable 'data' is listed in the Workspace interface panel (top right) with a little matrix icon next to it. See that the size of the matrix is 345x7 corresponding to the 345 rows and 7 comma-separated columns in the original text file.

4. Understanding the data

The data concerns the diagnosis of liver disease. Seven bits of information (7 columns) have been recorded about 345 patients (345 rows). We will talk about there being 345 *samples* and each sample having 7 *features*. The data in the 7 columns has the following meaning,

1. mean corpuscular volume
2. alkaline phosphotase
3. alamine aminotransferase
4. aspartate aminotransferase
5. gamma-glutamyl transpeptidase
6. number of half-pint equivalents of alcoholic beverages drunk per day
7. Class label (1=healthy; 2=diseased)

The first five are results from blood tests (we do not need to worry about their precise meaning), the sixth concerns the patient's alcohol consumption. The final column is a 'class label'. A 1 in this column means the patient is healthy and a 2 means the patient has liver disease. Full information about the data can be found here, http://archive.ics.uci.edu/ml/datasets/Liver+Disorders

If designing a medical diagnostic test, our goal would be to try and predict the class label in the seventh column (healthy vs. diseased) given the patient data recorded in the first 6 columns (the blood tests and alcohol consumption), i.e., it is a classification task (data in, label out).

5. Examining the data

Let us now look at one of the features in more detail. We can extract the *n*th column from a matrix M and store it in a vector, v, using the MATLAB syntax,

```
    v = M(:, n);
```

So to extract the information about alcohol consumption (feature 6) type

```
    drinks = data(:, 6);
```

We can now plot this data using the command,

```
    plot(drinks);
```

The plot generated will have the sample number along the x-axis (i.e. patient 1 to patient 345) and their alcohol consumption on the y-axis. This is not a very clear way of viewing the data. It would be more appropriate to use a histogram ("a bar chart"). This can be easily done by typing,

```
    hist(drinks);
```

Unfortunately by default the histogram has 10 bars which is not enough to display the data precisely. To generate a histogram with a bar for each of 0, 1, 2, … 20 units of alcohol type,

```
    hist(drinks, -0.5:1:20.5);
```

Type `help hist` to understand the meaning of the 2nd parameter.

You can make the figure a little clearer by changing the range of the axes, try,

```
    axis([-1,20,0,130]);
```

(take care with the square and round brackets). Type `help axis' for an explanation.

So we can now see that the people in the dataset are drinking between 0 and 20 units of alcohol ***a day!*** With the mode (most popular) being 1 unit. (Where would you lie on this distribution?)

To calculate the average alcohol consumption use the command 'mean(drinks)'

6. Using the class label

In the previous section we looked at the distribution of alcohol consumption for all 345 people. We now want to look at separate histograms for healthy and diseased people. Remember the class label is in column 7. We can make MATLAB test whether the entries in a column match a specific number, say for example 1, by using syntax like,

```
    data(:,7)==1
```

Take care to note that it is == and not = (i.e. the same as for comparisons in Java).

The results are returned as a vector of 1's (true) and 0's (false). This single line of code effectively performed 345 comparisons and returned 345 results all in one go.

We can store these results in a variable,

```
wellPeople = data(:,7)==1;
```

We can use these Booleans to select rows belonging to the healthy people from a column of our matrix, e.g. to select the column 6 data for just the healthy people,

```
wellDrinks = data(wellPeople,  6);
```

or putting it all in one line,

```
wellDrinks = data( data(:,7)==1, 6);
```

and similarly,

```
illDrinks  = data( data(:,7)==2, 6);
```

Now we want to compare the histograms for these two classes. We can place two plots in the same window using the 'subplot' command. Type `` `help subplot` `` for details.

```
subplot(2,1,1);
hist(wellDrinks, -0.5:1:20.5);
axis([-1,20,0,70]);
subplot(2,1,2);
hist(illDrinks, -0.5:1:20.5);
axis([-1,20,0,70]);
```

Compare the two histograms. Are you surprised by how they appear? Is alcohol consumption by itself a good predictor of liver disease?

7. Writing scripts

So far we have been typing directly into the command window. This can quickly become tedious when you want to repeat commands, or if you make a mistake and need to start again. The solution is to save commands in text files called *scripts* and then to execute these files.

MATLAB has a very convenient built-in script editor. Click on the '*New Script'* icon on the far left hand side of the tool bar. This will open a new window that is basically a glorified text editor.

Type some of the commands you have been using (e.g. the histogram and axis setting commands) into this window. You have written a MATLAB script. You can run this script by clicking on the `*Run'* icon in the script editor toolbar (green arrow). As you haven't saved it yet MATLAB will ask for a file name. It will then save the script as a text file with the extension .m. (You should see it appear in the folder contents pane of the main MATLAB interface). The script will run as soon as the save is complete. You can now easily edit the commands and click run to execute again.

Write a script that will execute all the commands so far from loading the data to comparing a pair of alcohol consumption histograms for the well and ill classes. Run the script and be sure it works. (Hint you can write it quickly by cutting and pasting from the command history).

Now edit the script so that it compares histograms for one of the other features. Hint: this means changing the '6' to some other column number from 1 to 5. You might want to change the names of variables too, e.g. rather than `wellDrinks` change it to `wellFeatureN` etc so that it makes more sense. Also you'll want to change the 2nd parameter of the histogram command and the axis settings so that they are appropriate for the new feature. Using 20 for the 2nd parameter of Histogram will produce a histogram with 20 bars evenly spaced across the data range which will generally produce a sufficiently detailed plot for seeing the shape of the distribution.

Which feature appears to be the best for distinguishing between the two classes? i.e. which feature generates a pair of histograms with the smallest overlap?

8. Looking at pairs of features using a scatter plot.

You'll probably find that no one feature is very useful in isolation. The classes will be better separated if we use more features. When looking at one dimension we can visualize the data distribution using a histogram. When looking at the distribution of a pair of features it is often better to use a *scatter plot*. With a scatter plot the pair of features are represented as a point on a 2-D plane, i.e. each sample is plotted on the plane at a position that represents the value of the sample's features.

Let's say that we want to look at feature 4 and feature 5. Again, we will separate the healthy people from the diseased people.

```
    wellFeatureX = data(data(:,7)==1, 4);
    wellFeatureY = data(data(:,7)==1, 5);
```

we can now plot these against each other using

```
    scatter(wellFeatureX, wellFeatureY);
```

Examine how the healthy patients are spread. We can now do the same for the diseased patients by changing the ==1 to ==2, i.e.

```
    illFeatureX = data(data(:,7)==2, 4); etc
```

When we plot the 2nd scatter plot it will replace the data from the 1st plot making it impossible to compare the two. To avoid this, after plotting the first scatter plot, type,

```
    hold on;
```

This stops the data being erased.

The problem now is that both sets of data are plotted using the same symbol so we can't see which is which. To use a different symbol and/or colour you can provide scatter with a 3rd argument.

For example, to use green crosses for the healthy people and red circles for the diseased people use,

```
    scatter(wellFeatureX, wellFeatureY, 'gx');
    hold on;
    scatter(illFeatureX, illFeatureY, 'ro');
```

How well are the two classes separated by this pair of features?

Now by writing scripts and using the scatter command try and find the pair of features that appear to visually separate the classes the best.

(Later we will learn how to use for-loops in MATLAB and will be able to do this very easily.)

9. Summary

MATLAB: We have been introduced to MATLAB and learnt a little about how to use the MATLAB interface. We have seen how MATLAB can store data in a matrix and how the data can be manipulated using a simple syntax. We have seen some of MATLAB's powerful plotting functions and used them to visualize some 1-D and 2-D data distributions using histograms and scatter plots.

Classification: We have been introduced to a simple classification task based on the diagnosis of liver disease using patient data. We have seen that for real world tasks it can be hard to find single features that clearly separate the classes we are interested in. We have seen that using more that one feature at a time can be a way of better separating the classes (i.e. the data in the scatter plots was less overlapped that the data in the histograms).

10. What next

MATLAB: In Friday's lecture we will take a more systematic look at how MATLAB handles matrices and vectors. We will show how it is possible to write programs in MATLAB and cover the usual language features (conditionals, loops and functions). We will introduce the concept of code 'vectorisation' and look at how to write MATLAB code that runs efficiently.

Classification: In the second part of the lecture we will introduce some mathematical tools that are later going to help us understand how to build effective classifiers, including basic probability theory and linear algebra.