(Concepts of) Machine Learning

Familiarising with MATLAB- part 2

After this activity you should be able to:

- Generate and manage data formulated as matrices;
- Implement simple algorithms to generate data;
- Work with plots.

Generate and manage data

Concatenation

Concatenation is the process of joining small matrices to make bigger ones. In fact, in session 1 you made your first matrix by concatenating its individual elements. The pair of square brackets, `[]`, is the concatenation operator.

Example:

Start with the 2-by-2 matrix *A*,

```
A= 1 2
   3 4
```

and form

```
B = [A  A+1; A+2  A+3]
```

The result is a 4-by-4 matrix, obtained by joining the four submatrices.

```
B= 1      2      2      3
   3      4      4      5
   3      4      4      5
   5      6      6      7
```

Deleting Rows and Columns

You can delete rows and columns from a matrix using just a pair of square brackets. Start with

```
X = B;
```

Then, to delete the second column of `X`, use

```
X(:,2) = []
```

This changes `X` to

```
X= 1      2      3
   3      4      5
   3      4      5
   5      6      7
```

If you delete a single element from a matrix, the result isn't a matrix anymore. So, expressions like

```
X(1,2) = [ ]
```

result in an error.

However, using a single subscript deletes a single element, or sequence of elements, and reshapes the remaining elements into a row vector. So:

```
X(2:2:11) = [ ]
```

results in

```
X = 1      3      2      4      3      5      7
```

Exercises

1) Create the following arrays using Matlab language:

(i) `A = all real numbers from 0.7 to 3 in increments of 0.5`

(ii) `B = 1,2,…,15`

(iii) `C = all even numbers between 8 and 15 of B`

(iv) Delete the first and last elements of `B`

(v) Replace all odd elements of `B` with `-1`

2) Perform the following calculations in Matlab command window

(i) `x = 0.01`

(ii) $y = x^2$

(iii) $z = \sqrt{x}$

(iv) `a = x*(y-z)`

Type: **`help sqrt`** and **`help ^`** to find details.

3) Generate each of the following arrays using Matlab language:

(i) array `a`, where `a(n)=0.1*n` and `n=1:10`

(ii) array `b`, where `b(n)=n*2 - 0.1*n -3*n` and `n=1:10`

(iii) array `c`, where `c =1/2,1/3,1/4,…1/20`

(iv) array `d`, where `d` $=1/2, 1/2^2, 1/2^3, 1/2^4, \ldots, 1/2^{20}$

Find out how to compute the sum of an array, and show that

$$\sum_{n=1}^{20} \frac{1}{2^n} \cong 1$$

Type: **`help sum`** to find details about this command.

Graphics

Use the help

Type:

```
help plot
```

Plotting functions or expressions

Create argument values:

```
x = start: increment: stop
```

The plotting function is:

```
plot(argument, f(argument))
```

Example:

`plot(t, sin(t).^2)` where `t =0:0.01:2*pi`

The figure window: adding and deleting curves

The default operation is that every use of plot command overrides previous plots.

Some useful operations include:

`hold on`: switches to add-a-new-plot mode
`hold off`: switches to replace mode
`clf`: clears the figure

Curve style

You can choose the line style, the marker and the colour that you will use to plot. These are inputted as additional parameters to the plot command. Use **`help plot`** to see how.

Example

`plot(t, sin(t).^2, 'r:')` where `t =0:0.01:2*pi`

Compare this graph with the previous one of the same function. Are there any differences?

Exercises

1) Implement in Matalb the functions:

```
u=sin(2*t)
v=sin(3*t)
```

for

```
t =0:0.01:2*pi.
```

Plot `u` versus `v`.

2) Plot `u` (alone).
Is this plot different from the previous one?

3) Assume that `u` is measured from `t=0` with increment of `0.1`.

```
(i) Plot u versus t.
```

(ii) Plot u^2.

4) Plot both `u` and `v` versus `t` on the same plot, using different colours and line-styles.
(Note: You have to use the command **`hold on`**, etc).