SLIIT
Discover Your Future

BSc in Information Technology
Specialising in Data Science

Semester 2

Lab Exercise 1-
Part A

IT3071 – Machine Learning and Optimization Methods 2021

Lab 1-Part A: Introduction to GNU Octave

Objective:

- To obtain a basic knowledge about GNU Octave Scientific Programming Language

Content:

- Introduction and Setup
- The Octave Environment
- Basic Operations
- Variables and data types
- Matrices
- Plotting Data
- Control Statements
- Functions
- More Information

1 Introduction and Setup

- GNU Octave is a high-level programming language like MATLAB
- It is also used for numerical computations.

- Download and install OCTAVE(GUI) on your own computer from below link:
  https://www.gnu.org/software/octave/download

- Alternatively, you can use online notebook version of OCTAVE from below link:
  https://cocalc.com/doc/octave.html

- To get help, "**type help**" or "**doc**"
- To get help on a specific command (=built-in function), type help command
  Examples: "**help size**", "**help plot**", "**help figure**", "**help inv**"
- To get help on the help system, type "**help help**"

2 The Octave Environment

- Navigation
  Octave uses a working directory just like terminal/command line.
  cd- Change Directory
  ls, dir -List current directory
  pwd- print working directory

- Loading and Saving Data

  - When you exit Octave, you lose all of the variables that you have created. If you need to quit Octave when you are in the middle of doing something, you can save your current session so that you can reload it later. If you type
    "save anyname"
  - It will save the entire workspace to a file called `anyname.mat` in your current directory (note that Octave automatically appends `.mat` to the end of the name you've given it). You can then quit Octave, and when you restart it, you can type
    "load anyname"
  - This will restore your previous workspace, and you can carry on from where you left off. You can also load and save specific variables. The format is
    save filename var1 var2 ...
  - For example, if you wanted to save the `deg` variable, to save calculating it from scratch (which admittedly is not very difficult in this case!), you could type
    "save degconv deg"
  - This will save the variable into a file called `degconv.mat` You can reload it by typing
    "load degconv"
  - Octave will also load data from text files, which is particularly useful if you want to plot or perform calculations on measurements from some other source. The text file should contain rows of space-separated numbers.
  - Octave's load function is very easy to use. Accepting many different data formats. (csv, tabulated, etc)
    load <filename> or load("<filename>")
  - This function loads the data into a vector or matrix. Octave can also save data very easily.
    save <filename> <variable>

  Example:

```
>> load single.dat
>> m=load("double.dat")
m =
  7.15071   1.49259
  2.37424   2.49971
  2.72140   7.38095
  3.96729   9.12556.....
```

```
>> v=rand(5,1)
v =
  0.941828
  0.543704
  0.014524
  0.291716
  0.336000

>> save mydata.dat v
>> v
v =
  0.941828
  0.543704
  0.014524
  0.291716
  0.336000
```

- Temporary Files
  Sometimes when dealing with large amounts of data, or if another program uses data from Octave you may need a temporary file.

  Temporary files are deleted when Octave closes.
  `f = tmpfile`
  `save f <variable>`
  `load f` (places data back into its old variable)

3 Basic Operations

- Arithmetic

Addition +
Subtraction –
Multiplication *
Division /
Exponent ** or ^
Element-by-element
Element-by-element addition .+
Element-by-element subtraction .-
Element-by-element multiplication .*
Element-by-element right division ./

```
>> 5+5
ans =  10
>> 5-9
ans = -4
>> 4*5
ans =  20
>> 1/3
ans =  0.33333
>> pi
ans =  3.1416
>> 8^2
ans =  64
>> [2 2]./[1.5 1]
ans =
   1.3333   2.0000
```

4 Variables and Data Types

Now we are going to talk about Octave variables.

- Variables
  What is a variable? It stores/hold a value for future use. We can assign a variable much like any other language
  ```
  A = 5;
  ```
  Octave will decide what type of variable it is on creation.
- Display Variables
  Simply type its name:
  ```
  >> a
  a = 4
  >> disp(a)
  ```
- Suppress Output
  Add a semicolon:
  ```
  >> a;
  >> sin(phi);
  ```
  Applies also to function calls.
- Variables have no permanent type. s = 3 followed by s = 'octave' is fine
- Use `who` (or the more detailed whos) to list the currently defined variables.
- Numerical Precision Variables are stored as double precision numbers in IEEE floating point format.

`realmin`	Smallest positive floating-point number: 2.23e-308
`realmax`	Largest positive floating-point number: 1.80e+308
`eps`	Relative precision: 2.22e-16

- Control Display of Float Variables. 'format' functions control the output format of the numeric values displayed in the Command Window. The format function affects only how numbers are displayed, not how OCTAVE computes or saves them.

`format short`	Fixed point format with 5 digits
`format long`	Fixed point format with 15 digits
`format short e`	Floating point format, 5 digits
`format long e`	Floating point format, 15 digits
`format short g`	Best of fixed or floating point with 5 digits (good choice)
`format long g`	Best of fixed or floating point with 15 digits

See help format for

- Float Variables

`ceil(x)`	Round to smallest integer not less than x
`floor(x)`	Round to largest integer not greater than x
`round(x)`	Round towards nearest integer
`fix(x)`	Round towards zero If x is a matrix, the functions are applied to each element of x.

5 Matrices

- There are lots of ways of defining vectors and matrices. Usually the easiest thing to do is to type the vector inside square brackets [].
  To delimit columns, use comma or space
  To delimit rows, use semicolon, for example

```
>> a=[1 4 5]
a =

   1   4   5

>> c=[4;7;10]
c =

   4
   7
   10
```

```
>> b=[2,1,0]
b =

   2   1   0

>>  A = [8, 2, 1; 3, -1,
4; 7, 6, -5]
A =

   8   2   1
   3  -1   4
   7   6  -5
```

- Creating a Matrix from Matrices

```
>> A = [1 1 1; 2 2 2];  B = [33; 33];
Column-wise
>> C = [A B]
C =
 1 1 1 33
 2 2 2 33
Row-wise:
>> D = [A; [44 44 44]]
D =
 1 1 1
 2 2 2
 44 44 44
```

- The colon notation
  A useful shortcut for constructing a vector of counting numbers is using the colon symbol ':', as in this example

```
>> e=2:6
e =
2 3 4 5 6
```

  The colon tells Octave to create a vector of numbers starting from the first number, and counting up to (and including) the second number. A third number may also be added between the two, making a : b : c. The middle number then specifies the increment between elements of the vector.

```
>> e=2:0.3:4
e =
2.0000 2.3000 2.6000 2.9000 3.2000 3.5000 3.8000
```

Note that if the increment step is such that it can't exactly reach the end number, as in this case, it generates all of the numbers which do not exceed it. The increment can also be negative, in which case it will count down to the end number.

- Get Size

`nr = size(A,1)`	Get number of rows of A
`nc = size(A,2)`	Get number of columns of A
`[nr nc] = size(A)`	Get both (remember order)
`l = length(A)`	Get whatever is bigger
`numel(A)`	Get number of elements in A
`isempty(A)`	Check if A is empty matrix []

- Matrix Operations

`B = 3*A`	Multiply by scalar
`C = A*B + X - D`	Add and multiply
`B = A'`	Transpose A
`B = inv(A)`	inv(A) Invert A
`s = v'*Q*v`	Mix vectors and matrices
`d = det(A)`	Determinant of A
`[v lambda] = eig(A)`	Eigenvalue decomposition
`[U S V] = svd(A)`	Sing. value decomposition

- Special Matrices

`zeros(M, N)`	Create a matrix where every element is zero. For a row vector of size n, set M = 1, N = n
`ones(M, N)`	Create a matrix where every element is one. For a row vector of size n, set M = 1, N = n
`linspace(x1, x2, N)`	Create a vector of N elements, evenly spaced between x1 and x2
`logspace(x1, x2, N)`	Create a vector of N elements, logarithmically spaced between 10^{x1} and 10^{x2}

6 Plotting Data

Octave has powerful facilities for plotting graphs via a second open-source program GNUPLOT. The basic command is `plot(x,y)`, where x and y are the co-ordinates

```
>> angles=[0:pi/3:2*pi]
>> y=sin(angles)
>> plot(angles,y)
```

- Frequent Commands

`clf`	Clear figure
`hold on`	Hold axes. Don't replace plot with new plot, superimpose plots
`grid on`	Add grid lines
`grid off`	Remove grid lines
`title('Exp1')`	Set title of figure window
`xlabel('time')`	Set label of x-axis
`ylabel('prob')`	Set label of y-axis

- Controlling Axes

`axis equal`	Set equal scales for x-/y-axes
`axis square`	Force a square aspect ratio
`axis tight`	Set axes to the limits of the data
`a = axis`	Return current axis limits [xmin xmax ymin ymax]
`axis([-1 1 2 5])`	Set axis limits (freeze axes)
`axis off`	Turn off tic marks
`box on`	Adds a box to the current axes
`box off`	Removes box

- Choosing Symbols and Colors

  In `plot(x,cos(x),'r+')` the format expression `'r+'` means red cross.
  There are a number of line styles and colors, see help `plot`.

  Example:

```
octave:1> x = linspace(0,2*pi,100);
octave:2> plot(x,cos(x),'r+',x,sin(x),'bx');
```

  produces this plot

- Plotting in 3D

`plot3`	Plot lines and points in 3d
`mesh`	3D mesh surface plot
`surf`	3D colored surface plot

  Most 2D plot commands have a 3D sibling. Check out, for example, `bar3, pie3, fill3, contour3, quiver3, scatter3, stem3`

7 Control Statements

Octave supports the usual loops and selection facilities.

- if Statement

```
if condition,
then-body;
elseif condition,
elseif-body;
else
else-body;
end
```

Example:

```
>> a=0; b=2;
>> if a > b
c=3
else
c=4
end

>> 1==2
ans =0
>> pi > exp(1) & sqrt(-1) == i
ans =1
```

- switch Statement

```
switch expression
 case label
 command-list;
 case label
 command-list;
 ... otherwise
 command-list;
end
```

Example:

```
>> a=1;
>>switch a
case 0
disp('a is zero');
case 1
disp('a is one');
otherwise
disp('a is not a binary digit');
end
```

- while Statement

```
while condition,
 body;
end
```

Example:

```
>> x=1;
>> while 1+x > 1
x = x/2;
end
>> x
x =
1.1102e-016
```

- for statement

```
for var = expression,
 body;
end
```

Example:

```
>> for n=1:5
nf(n) = factorial(n);
end
>> disp(nf)
1 2 6 24 120
```

8 Functions

Scripts in Octave let you write simple programs, but more powerful than scripts are user-defined functions.
In its simplest form, the definition of a function named name looks like this:

```
function name
 body
end
```

- .m files
  Each function need to go into it own .m file of the same name. .m files are just text files that hold Octave code to be run.
  Eg. minimum(x) needs to go in a file called minimum.m

Example:
There is already an inbuilt min function but for learning purposes lets make our own minimum function! File name: mini.m

```
function ret = minimum(x)
      ret = x(1);
      for i = 2:length(x),
            if x(i) < ret,
                  ret = x(i);
            end;
      end;
endfunction
```

- Multiple return values
  Octave supports returning more than one variable from a function. Code file name: minmax.m

  Example:
  Let's combine a minimum and maximum function into one function called minmax.

```
function [low,high] = minmax(x)
      low = x(1);
      high = x(1);
      for i = 2:length(x),
            if x(i) < low,
                  low = x(i);
            elseif x(i) > high,
                  high = x(i);
            end;
      end;
endfunction
```

8 More Information

Full Octave online documentation:
http://www.octave.org
➢ Docs
➢ 575 page manual
(directly: www.gnu.org/software/octave/doc/interpreter)

In class task 01

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{matrix}$$

1. Instantiate the above matrix using the octave command.
2. What are the outputs from the following commands?
   a) A(3,1)
   b) A(1,3)
   c) A(:,4)
   d) A(1,:)
   e) A(1,1:3)
   f) A(1:4,5)
   g) A(1:3,1:3)
   h) A(1:2:5,1:2:5)
   i) A(1:3,:)=[]

In class task 02

Octave has a special way of handling polynomials. As an example, consider the third order polynomial f which is a function of x

$$2x^3 + 10.1x^2 + 6 = \mathrm{f(x)}$$

We can represent this polynomial via a vector containing the coefficients:

$$c = [2\ 10.1\ 0\ 6]$$

We can now evaluate the function range or value for x = 0 by using polyval: Also, we can calculate the range using a vector as input. For example `f=polyval(c,x);`

1. In this task plot the polynomial, f, given in Equation above the interval x ∈ [–5.5; 1] (HINT: `x=[-5.5:0.1:1]`).

   There are some things that do not look quite satisfactory in the answer 1 plot figure:
   - The axes are not right, for example, the x axis starts from -6, not -5.5.
   - The graph and the window box lines are too thin.
   - The axes are not labelled.
   - The numbers on the axes are too small.
2. Fix the above errors. (HINT: `plot (x, y,fmt, property, value, ...)` and `set(handle, property, value, ...)`)
3. Plot another polynomial in the same interval on the same window.

$$2x^3 + 10.1x^2 - 10.1x + 6 = \mathrm{f(x)}$$

4. Save the figure window to an image format named Plot1.