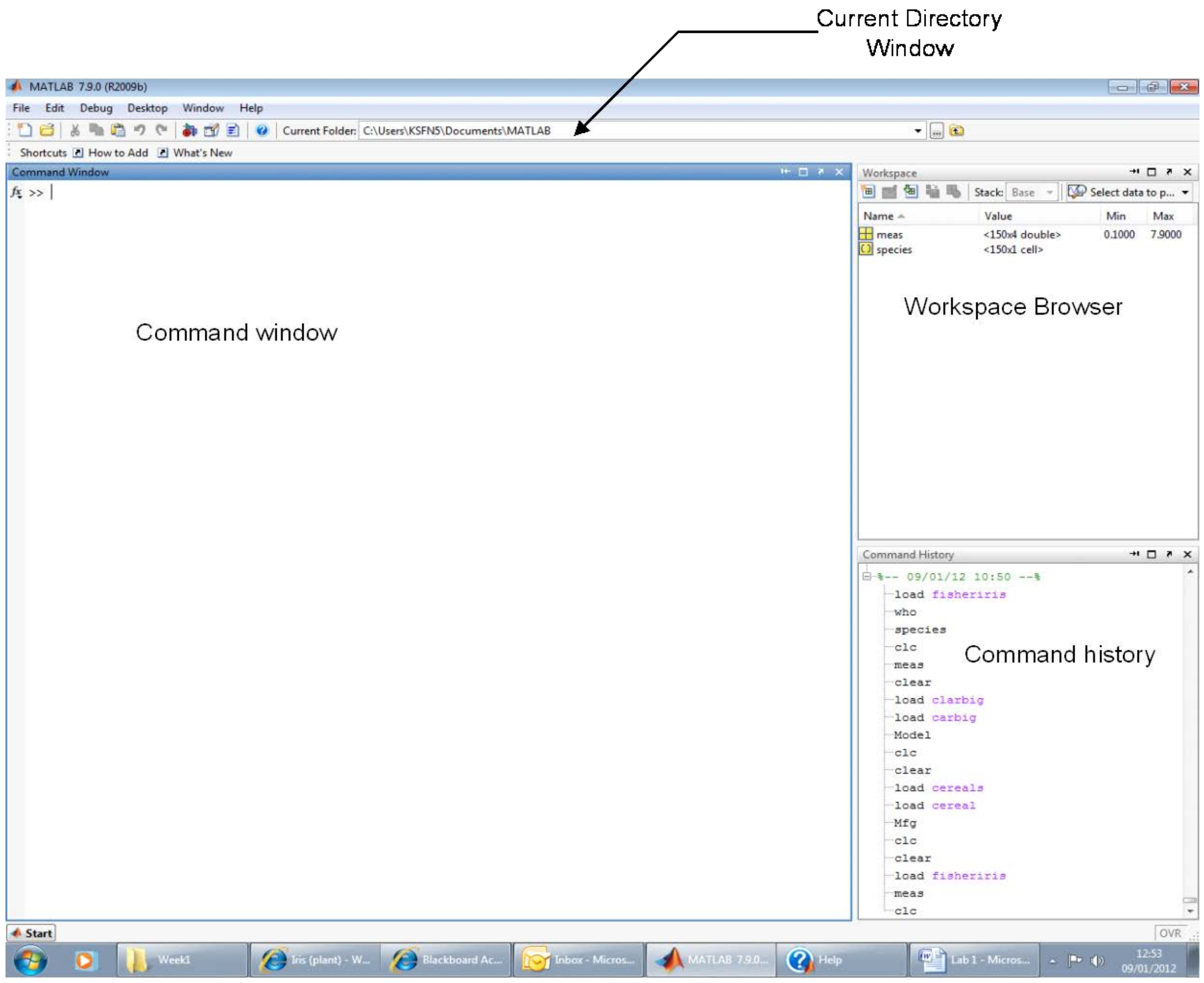
CM0669 Machine Learning and Computer Vision

Lab 1 Introduction to MATLAB Programming/ Machine learning and linear classifiers

1. Matlab environnment

Matlab is a high-performance language for technical computing. It has particularly been developed for research purposes. The Matlab desktop (figure below) is the main Matlab application window.



Name	Value	Min	Max
meas	<150x4 double>	0.1000	7.9000
species	<150x1 cell>		

```
%-- 09/01/12 10:50 --%
load fisheriris
who
species
clc
meas
clear
load clarbig
load carbig
Model
clc
clear
load cereals
load cereal
Mfg
clc
clear
load fisheriris
meas
clc
```

Normally, the desktop contains four sub-windows: the command window, the workspace browser, the current directory, and the command history.

The **command window** is where the user types MATLAB commands and expressions at the prompt (>>) and where the output of those commands are displayed. It can also be used to run a Matlab programme by just typing its name at the prompt.

The **current directory** tab shows the path for the Matlab codes to be executed. You will need to change the path accordingly to run your own Matlab programme.

Matlab defines the workspace as the set of variables that the user creates in a work session. The **workspace browser** shows these variables and some information about them.

The command history window contains a record of the commands a user has entered in the command window.

2. First programme in Matlab

Before you start, create a folder named ‘Week1’ in your ‘U’ drive and change the current directory accordingly. The Matlab files that you will create (or download) should be saved in ‘Week1’. Through this example, you will see the advantage of using Matlab compared to other High level programming languages. In Matlab, each variable is considered a **Matrix**. For instance, a real value is a matrix of one row and one column (size equals (1,1)). Unlike the programming languages you have seen so far, Matlab does not require the type of variables to be declared before they are used. In this example, the task is to write a Matlab code which calculates the sum of squared integers from 1 to N (N to be set by the user).

$$S = \sum_{i=1}^{N} i^2 = \sum_{i=1}^{N} (i \times i) = 1 + 2^2 + 3^2 + \cdots + N^2$$

There are two alternatives to implement the above equation:

A. **Loop-based method** (using ‘for’ loop like Java or C++)
   Here is the code:

```
N=100;  % This is a comment. Comments are preceded by ‘%’. Set N to 100.
S=0;      %  initialise the output with zero. Each line is terminated by a semi-colon.
for i=1:1:N,
     S=S+i^2;   % add the new squared integer to the previous value of S.
end
disp('The output is '); S   % display the result. Note S is not terminated by ‘;’
```

Create a new file named ‘example1’ using Matlab desktop and save it in the folder you have created previously (Week1). Copy the code above, and save the file through the Matlab editor. To run the code, you either:

- Click on ‘debug’ then ‘run’, or
- Go to the Command window, type the file name and press enter.

In a Matlab programme, if a line is not terminated by a semi-colon, the value of the corresponding variable will be displayed.

B. **Matrix-based method** (using a vector)

Here is the code:

```
N=100;  % This is a comment. Comments are preceded by '%'. Set N to 100.
Vector=1:1:N; % define a vector of incrementing integers from 1 to N.
S=sum(Vector.^2);
disp('The output is '); S   % display the result. Note S is not terminated by ';'
```

In this method, 'Vector' is an array containing integer values from 1 to N.

That is, Vector={1,2,3,4, ..., N}. The function 'sum' is a built-in function in Matlab which calculates the sum of the numbers in an array. For more info, type

```
>> help sum
```

Note the use of **dot** in (Vector**.**^2) which tells Matlab that the square operation (^2) is applied to an array of elements.

Create a new file named 'example2' using Matlab desktop and save it in the folder you have created previously (Week1). Copy the code above, and save the file. Run the code and compare the result with the previous one.

Remember that all the variables you have used 'S', 'Vector', 'i', and 'N' are currently known to Matlab. You should be able to see them in the workspace browser window. If you type the name of any variable you should get its value. For instance,

```
>> N

N=

    100
```

To check the variables which are currently known to Matlab, type:

```
>> who

Your variables are:


N      S      Vector   i
```

To clear one variable from the workspace, say N, type

```
>> clear N;
```

This means that the Matlab workspace does not know ‘N’ anymore. Note that this variable has disappeared from the Matlab workspace window. To clear all variables, type:

```
>>clear all;
```

To clear the Matlab command window, type

```
>>clc
```

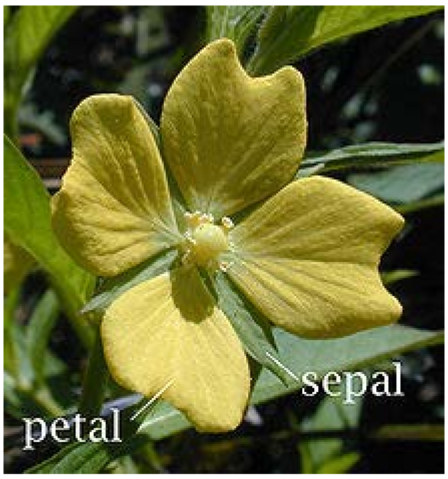
Make sure you know the difference between ‘clc’ and ‘clear’.

3. Linear classification (gradient descendent algorithm)

3.1. Fisher's Iris Data

Fisher's iris data consists of measurements on the sepal length, sepal width, petal length, and petal width for 150 iris specimens (samples). There are 50 specimens from each of three species.

Iris Plant



Petal and Sepal for a flower

Use the Matlab Command window to load the data, type

```
>> load fisheriris
```

Now Matlab has read the data from a file ‘fisheriris.mat’. To see the variables which contain this data, type

```
>> who
```

‘meas’ consists of measurements on the sepal length (first column), sepal width (second column), petal length (third column), and petal width (fourth column). The expected species for each entry (row) is given by the variable ‘species’. To check the data for ‘meas’, type

```
>>meas
```

For species, type

```
>> species
```

As you can see, the variable ‘species’ is a column vector of 150 values representing three different classes ‘setosa’, ‘versicolor’, and ‘virginica’. Each class depends on the value of features given by the variable ‘meas’. For instance, if

$$\left.\begin{array}{r} sepal\ length = 5.1000 \\ sepal\ width = 3.5000 \\ petal\ length = 1.4000 \\ petal\ width = 0.2000 \end{array}\right\} then\ species = 'setosa'$$

‘meas’ is a table (seen as a matrix in Matlab) with four columns and 150 rows. ‘species’ is column of 150 values (seen as a matrix of size (150,1)). To check the size of the variables, type

```
>> [rows_meas,columns_meas]=size(meas)
```

```
>>[rows_species,columns_species]=size(species)
```

To see how the sepal measurements differ between species, you can use the two columns containing sepal measurements (first and second columns). Type

```
>> gscatter(meas(:,1), meas(:,2), species,'rgb','osd'); xlabel('Sepal length');
ylabel('Sepal width');
```

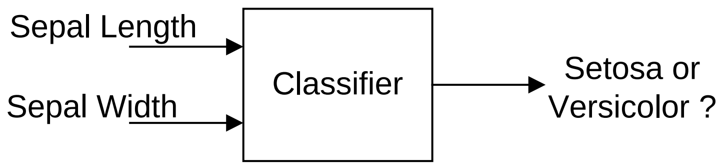
gscatter(X,Y,G) creates a scatter plot of the vectors X and Y grouped by G. In the previous example, meas(:,1) represents the first column of ‘meas’ (sepal length) and meas(:,2) represents the second column of ‘meas’ (sepal width). Becasue there are three classes, ‘rgb’ is used to plot the first one in red, the second one in green and the third one in blue. Similarly, ‘osd’ is used to plot the first class ‘setosa’ with a circle (‘o’), the second class (‘versicolor’) with a square (‘s’) and the third one (‘virginica’) with a diamond (‘d’). ‘xlabel’ and ‘ylabel’ are used to insert a description for the corresponding axes. Note that all command lines that you have seen in the Matlab command window can be used in a Matlab script (Matlab programme) which can be run as seen in **section 2**.

In this week, we consider the classification of the data in two classes. That is, specimens for two classes only are used for training and testing our **linear classifier**.

3.2. Gradient-descent linear classification

Download the matlab codes 'SL_SW_out_SE_VERS.m', 'PL_PW_out_SE_VERS.m', 'SL_SW_PL_PW_out_SE_VERS.m', 'SL_SW_out_VERS_VIRG.m', 'PL_PW_out_VERS_VIRG.m', 'SL_SW_PL_PW_out_VERS_VIRG.m' in folder 'Week1'.

Each Matlab code implements a linear classifier using the gradient-descent rule for the training. However, they differ in terms of the input and output used. For instance, 'SL_SW_out_SE_VERS.m' is a classifier using as input **S**epal **L**ength and **S**epal **W**idth and as output the '**SE**tosa' and '**VERS**icolor' class labels (See figure below)



For the evaluation of each classifier, 80 samples (specimens) have been used for the training while the testing is performed on 20 samples.

A. Open up each Matlab code using the Matlab workspace and understand its content (note that each Matlab code is well commented.).
B. Run the codes and complete the results in the table given below.
C. Compare and analyse the results.
D. Change the initial parameters of the classifier in each Matlab code and then run it again. Compare the new training/testing errors with the previous ones. Interpret the results.

Input				Output			Evaluation	
Sepal Length	Sepal Width	Petal Length	Petal Width	Setosa	Versicolor	Virginica	Training Error (%)	Testing Error (%)
Yes	Yes	No	No	Yes	Yes	No		
No	No	Yes	Yes	Yes	Yes	No		
Yes	Yes	Yes	Yes	Yes	Yes	No		
Yes	Yes	No	No	No	Yes	Yes		
No	No	Yes	Yes	No	Yes	Yes		
Yes	Yes	Yes	Yes	No	Yes	Yes		

Exercise (optional)

Create new Matlab codes (three codes) from the existing ones so that you can fill in the following table:

Input				Output			Evaluation	
Sepal Length	Sepal Width	Petal Length	Petal Width	Setosa	Versicolor	Virginica	Training Error (%)	Testing Error (%)
Yes	Yes	No	No	Yes	No	Yes		
No	No	Yes	Yes	Yes	No	Yes		
Yes	Yes	Yes	Yes	Yes	No	Yes		