CS342 Machine Learning: Lab #1

Labs on Week 1 of Term 2

Week 15

Instructor:
Dr Theo Damoulas (T.Damoulas@warwick.ac.uk)

Tutors:
Helen McKay (H.McKay@warwick.ac.uk),
Joe Meagher (J.Meagher@warwick.ac.uk)
Karla Monterrubio-Gomez (K.Monterrubio-Gomez@warwick.ac.uk),
Jevgenij Gamper (J.Gamper@warwick.ac.uk)

In this first Lab we will familiarise ourselves with Python and some of the main libraries, and then run a simple linear regression with ordinary least squares on the Olympic dataset from the R&G book. Refer to the module website `https://www2.warwick.ac.uk/fac/sci/dcs/teaching/modules/cs342/` for the dataset, related background material, and module slides. If you are already comfortable with Python please proceed to the Machine Learning component of the Lab. If you do not manage to complete the Lab in the pre-designated hour then please complete it in your own time as it will help you better understand some of the course material.

Introduction

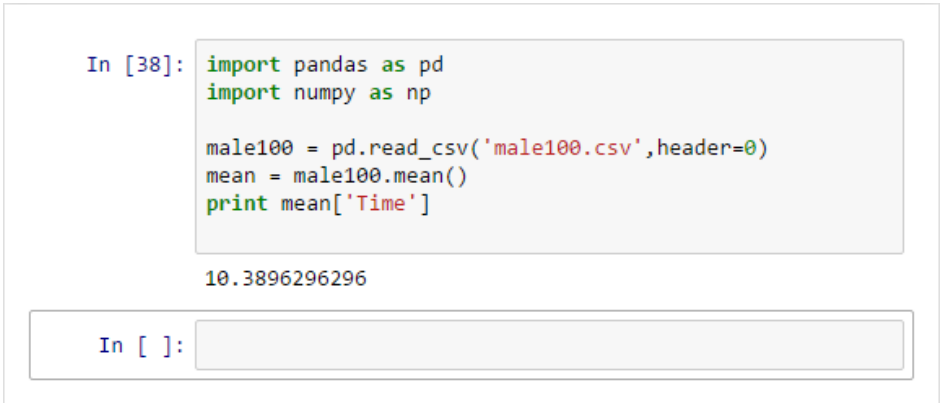
All the machines in the Lab have the main Python libraries and environment you will need today. If you are using your own machine, you can install the **Anaconda** distribution.`https://www.continuum.io/downloads` which includes most of the libraries we will need for the Labs.

You will need an editor/IDE to develop your codes and if you don't have a preference we recommend using the **iPython Notebook** which is available on all DCS machines. The Jupyter iPython Notebook can be launched by typing any of following 3 commands in a Terminal (Linux or Mac) or Command Prompt (Windows): (**Note: To launch on DCS machines use jupyter-notebook**).

```
ipython notebook or jupyter notebook or jupyter-notebook
```

To start a new notebook select the **'New'** button in the top right corner and select Notebooks-**Python2**.

The following image shows a basic interface of iPython Notebook. The interface splits into cells. Python code can be programmed in each cell. You can choose to run your python code in a single cell by (**Shift-Enter** / **Ctrl-Enter**) or the whole notebook by clicking on the menu Cell → Run All.



You can load/edit/run/save python files(.py) into an iPython notebook cell by calling magic commands. Here are some useful magic commands:

- **Load**: %load filename.py
- **Write/Save**: %writefile myfile.py
- **Run**: %run myfile.py
- **Inline Plotting**: %matplotlib inline

Python Quick Guide

A good reference for Python is this online book: `http://www.greenteapress.com/thinkpython/html/index.html`. Below are a couple of general Python coding exercises to get you going. If you are not comfortable with Python try implementing these first before proceeding to the machine learning lab material.

Common errors: If copying and pasting this code, please make sure the quotation marks are correct in your python script.

Example 1

Example 1 demonstrates basic Python syntax. Unlike many other languages which use a semicolon(;) to indicate the end of the statement, Python has no mandatory statement termination characters, however it is whitespace sensitive (i.e. you must indent your code properly). In addition, Python is dynamically, implicitly typed (i.e. you don't have to declare variables). Comments in Python start with the hash character # , and extend to the end of the physical line.

Listing 1: Basic Python Syntax

```
#Declearing variable type is not required
a_string = "hello, world"
an_integer = 12
a_float = 3.14
a_boolean = True

#To print a constant or variable, use commas to print several items,
print a_string, an_integer, a_float, a_boolean, "\n"

#A long statement may be split into different lines with a backslash:
print 'a long statement may be split using backslash', \
    'this is still the same statement', "\n"

#Indentation Example
x = 10
if x == 10:
   print ('x has a value of 10')
else:
   print ('x does NOT have a value of 10')
```

Example 2

Listing 2 introduces the use of lists and dictionaries as data structures in Python. These are the building blocks for various other Python data structures, including DataFrames which can be used to store instances of data (similar to records in a database).

Listing 2: Examples of lists and dictionaries

```
myList = [1,2,3,4]

#range() creates a list of numbers between 0 and 10 in increments of 1
myList2 = range(0,10,1) #listof numbers 0 to 10

#this appends the object myList to the end of myList2 - i.e. the last element of
#myList2 is a list
myList2.append(myList)
print myList2

#remove(x) removes the first element in the list that matches x
myList2.remove(myList)

#to add each element of myList to myList2, we can use a for loop
for x in myList:
    myList2.append(x)
print myList2

#len(x) returns the length of x
print len(myList2)

#the sort function can take additional parameters such as a key,
#ascending/decending etc
myList2.sort()

#slicing lists - allows you to select segments of the list
print myList2[2:4]
print myList2[:3] # first 3 elements

#dict contain key value pairs
myDict = {'a':'hello','b':'world'}
print myDict['a']
print myDict['b']
```

Example 3

Listing 3 shows how functions can be defined within Python scripts.

Listing 3: Example Python function

```
#This Fuction output the result 1+1
def onePlusOne():
     return 1+1

#This function prints a passed string
def printme( str ):
   print str
   return

#Function Return sum of input values a and b
def add (a,b):
     return a+b

#Call Function
x = onePlusOne()
y = add(1,2)

#native print fucntion == printme()function
print x
printme(y)
```

Pandas Quick Guide

Common errors: When loading data (e.g. read_csv) make sure you use the correct (relative) file path to the data! You can download the data files from the module webpage.

Example 4

Pandas is a Python library that provides easy to use data structures and data analysis tools. As Pandas is built on top of the Numpy package, both Pandas and Numpy packages are usually required to be imported for complete functionality. Example 4 below demonstrates how to import and export dataset in pandas. All the data imported will be stored into a data structure called a DataFrame. For more details on Pandas functions, see the API Reference at `http://pandas.pydata.org/pandas-docs/stable/api.html`.

Listing 4: Introduction of Pandas

```
#imports the Pandas and Numpy libraries and gives aliases pd/np.
import pandas as pd
import numpy as np

#import csv file. If file contains header row, set header to the line 0
#the import dataset will be stored into DataFrame (male100)
male100 = pd.read_csv('male100.csv',header=0)
print male100

#output male100 dataset to demo.csv file
male100.to_csv('demo.csv')
```

The table below is the output of printing the Male100 DataFrame. This dataset contains Olympics men's 100 metres winning times for each Olympic year. In later exercises, you will need to evaluate this dataset, which is available on the module webpage.

Table 1: Data Structure of Male100.csv

	Year	Time
0	1896	12.00
1	1900	11.00
2	1904	11.00
3	1906	11.20
4	1908	10.80
5	1912	10.80
6	1920	10.80
...	...	...

Example 5

Pre-processing a dataset is often crucial in order to obtain a more accurate ML model. The following example provides some basic data processing functions available in Pandas. In general it is good practice to perform pre-processing stages on a copy of the original DataFrame so that the original data remains unaltered. To do this use *copydf = df.copy()* as *copydf = df* only passes by reference, therefore changes made to *copydf* will effect the original DataFrame.

Example 5 continued on next page...

Listing 5: Data Processing

```
import pandas as pd
import numpy as np

male100 = pd.read_csv('male100.csv',header=0)

#Each Column can be extracted by "daraframe name[column name]"
print male100['Time'], "\n"

#Calculate mean and standard deviation
#other common functions also available: max(), median()...
mean = male100['Time'].mean()
std = male100['Time'].std()

#To get some basic statistics, we can use the describe() method:
print male100['Time'].describe(), "\n"

print mean, std
```

Example 6

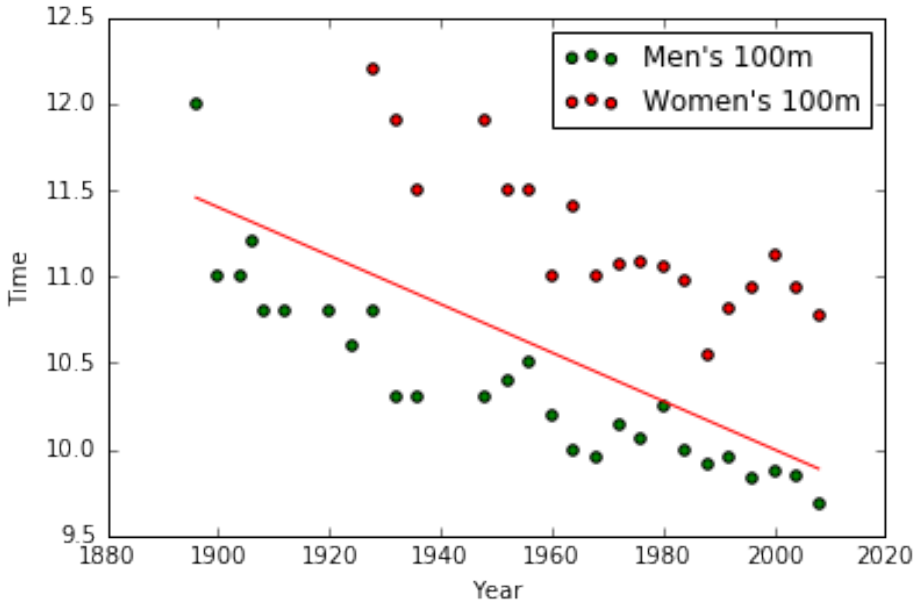
Example 6 introduces basic graph plotting. If you are using iPython Notebook, a Magic Line *%matplotlib inline* can be used to activate the inline graph display.

Listing 6: Basic Plotting

```
#This magic command is used to activated the inline graph display
%matplotlib inline

import pandas as pd
import numpy as np

#import 'matplotlib.pyplot' to plot a simple stright line
import matplotlib.pyplot as plt

male100 = pd.read_csv('male100.csv',header=0)
female100 = pd.read_csv('female100.csv',header=0)

#Basic pandas plotting
male100.plot(x=0,y=1, kind ='scatter', color='g', marker='v', label="Mens 100m")

#Simplified Version
male100.plot.scatter(0,1, color='g', label="Mens 100m")

#Two different dataset in one graph
ax = male100.plot(x=0,y=1, kind ='scatter', color='g', label="Mens 100m")
female100.plot(x=0,y=1, kind ='scatter', color='r', label="Womens 100m", ax = ax)

#we can use plt(imported from matplotlib) to plot a simple graph
#define a graph with repect to the all Olympic Years in male100.csv
y = -0.014*male100['Year']+38
plt.plot(male100['Year'],y,'r-',color = 'r')
```

Example 6 continued on next page...

Display Result



The above code uses the function *DataFrame.plot(x, y, kind, color, label, ax)*. The first two parameters indicate which columns of the DataFrame will be used for the x and y axes. *color* is used to assign a colour to data points, *marker* defines their shape, and *label* gives the graph a name. *kind* allows the plotting method to be set, the default is line, however, many other plotting methods are available, including:

- *bar* or *barh* for bar plots
- *hist* for histogram
- *box* for boxplot
- *kde* or *density* for density plots
- *area* for area plots
- *scatter* for scatter plots
- *hexbin* for hexagonal bin plots
- *pie* for pie plots

A full list of parameters that can be used within *DataFrame.plot()* can be found at `http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html`. Additional colour options are also available `http://matplotlib.org/api/colors_api.html`, as are marker shapes `http://matplotlib.org/api/markers_api.html`. In order to plot different DataFrames in the same figure, we can use the *ax* variable, for example:

```
ax = df1.plot()
df2.plot(ax=ax)
```