Department of Computer Engineering
University of Peradeniya

Data Mining and Machine Learning
Lab 02

June 27, 2017

1 Introduction

Making plots and visualizations are one of the most important tasks in data mining and machine learning. It may be a part of the exploratory process; for example, identify outliers, data transformations, or coming up with ideas for models. Matplotlib is a Python package for 2D plotting which provides a wide variety of plot types such as lines, bars, pie charts and histograms. Futher, it uses multiple window toolkits such as GTK+, wxWidgets and Qt.

1.1 Importing Matplotlib

```
import matplotlib
import matplotlib.pyplot as plt
```

1.2 Figures and Subplots

We start with basic plotting and lay down the fundamentals of plotting in the following sections. x and y-axes, x and y tickers, x and y tick labels, and plotting area are the basic elements which contains in a plot which is shown in Fig. 1

1.2.1 Matplotlib Configuration and Global Setting

```
# TODO How to switch backends in matplotlib?
# matplotlib.use('Agg')
# TODO Why do we need to change the backend of matplotlib?
# GTK GTKAgg GTKCairo GTK3Agg GTK3Cairo
matplotlib.rc('xtick', labelsize=30)
matplotlib.rc('ytick', labelsize=30)
matplotlib.rc('axes', titlesize=30)
matplotlib.rc('legend', fontsize=30)
```

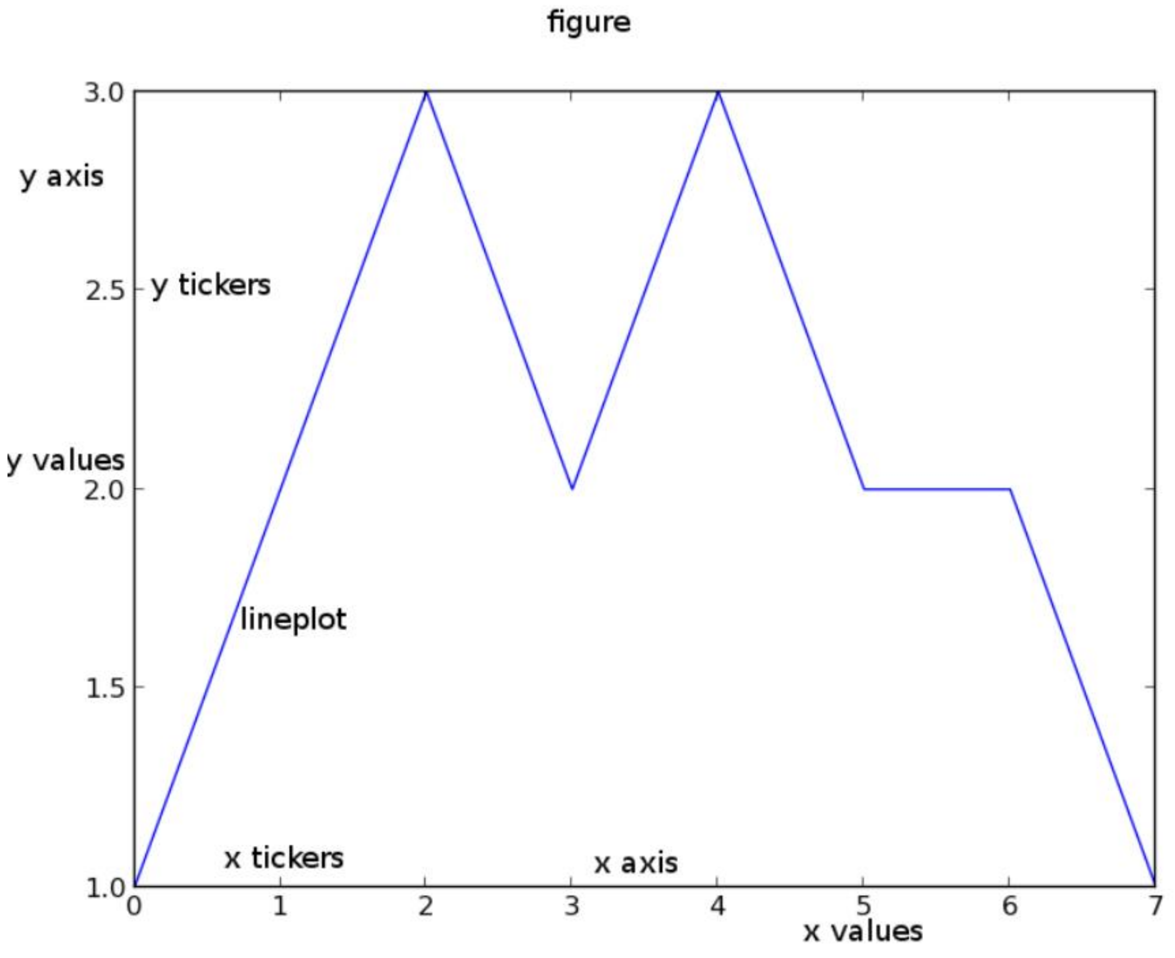

Figure 1: Plot Anatomy

1.2.2 Create a Figure

```
fig = plt.figure(figsize=(10, 10))
fig.subplots_adjust(hspace=.5) # TODO Is there any reason for
# setting this?
axes = plt.subplot(5,1, 1)  # row-col-subplot_number
axes = plt.subplot(511) # this is another way of initializing subplots
```

1.2.3 Colors, Markers, and Line Styles

```
x = np.linspace(0, 10, 100)
y = np.cos(x)
axes.plot(x,y,linewidth=4.0, ls='--', color='r', marker="o")
```

1.2.4 Ticks, Labels, Title and Legends

```
axes.legend("cos(x)",loc="upper right")
axes.set_title("Title for (5,1) subplot")
axes.set_xlabel('xlabel')
axes.set_ylabel('ylabel')

axes.set_autoscaley_on(False)
axes.set_ylim([0,1])
axes.xaxis.set(ticks=range(1,10))

fig.text(0.5, 0.04, 'common x label', ha='center', fontsize=15)
fig.text(0.04, 0.5, 'common y label', va='center', rotation=
'vertical', fontsize=15)
```

1.2.5 Saving Plots to File

```
plt.savefig('cosine.png', dpi=400, bbox_inches='tight')
# dpi, which controls the dots-per-inch resolution, and bbox_inches,
# which can trim the whitespace around the actual figure
```

1.3 Try Out

Let us try to visualize the path of a random walker problem done in the Lab 01. In your random walker implementation, add a new method which can be used to retrieve the position vector(Pos_{v}). It should include random walker current position corresponds to each step. The following code segment can be used to visualize the position vector.

```
  plt.plot(Pos_v)
  plt.show()
```

Then create a figure which contains three subplots such that each subplot should have its own legend, title and labeling for x & y axes. Each plot should contain the following subplots.

1. subplot 01: Pos_{v} of random random walker
2. subplot 02: The histogram of Pos_{v}

   ```
     plt.hist(Pos_v, bins=20, color='r')
   ```

3. subplot 03: A scatter plot can be used to plot correlation between two vectors. Run the random walker twice and store position vector corresponding to each run (Pos_{v1} and Pos_{v2}). Use the following code segment to get the correlation between Pos_{v1} and Pos_{v2}.

   ```
   plt.scatter(Pos_v1,Pos_v2, color='green', alpha=0.3,
   edgecolors='grey')
   ```

Extra : instead of positioning a legend on each subplot think about how to add a common legend for the whole plot.

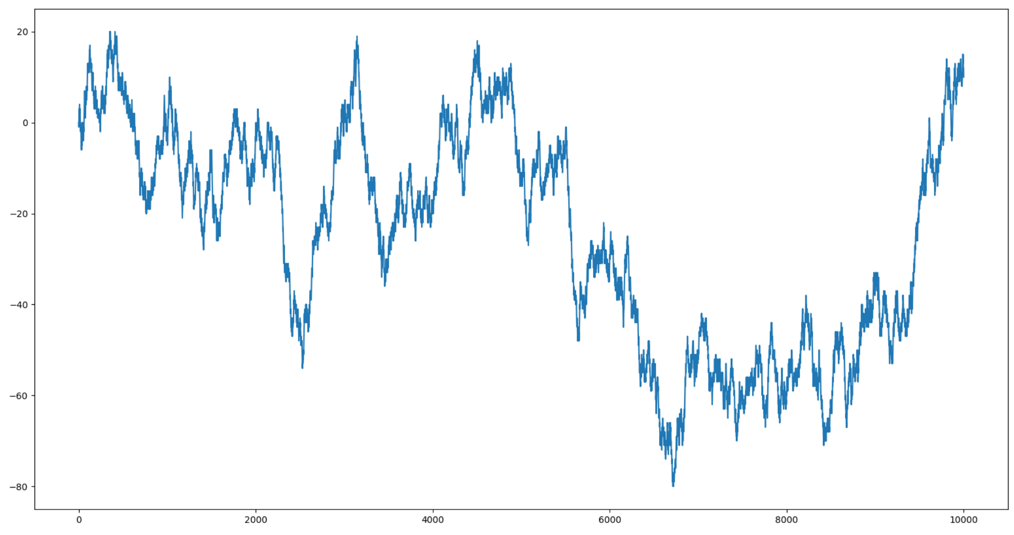

Figure 2: Example path of a random walker

2 Test Your Code

2.1 RandomWalk Class

```
class RandomWalk():
    def __init__(self, start_position, steps):
        self.start_position = start_position
        self.steps = steps

    def walk(self):
        return walked_path
```

2.2 Writing Test Cases

```
import unittest

class RandomWalkTest(unittest.TestCase):

    def setUp(self):
        """ Setting up for the test """
        print "RandomWalk:setUp_:begin"
        self.random_walker = RandomWalk(0, 500)

    def tearDown(self):
        """Cleaning up after the test"""
        print "RandomWalk:tearDown_:begin"
    def testStartPosition(self):
        self.assertEqual(self.random_walker.start_position, 0)
```

2.3 Run Test Cases

```
if __name__ == '__main__':
    unittest.main()
```

3 Lab Exercise

In the field of solving an unconstrained optimization problem, one of the most well-known method is the Gradient Descent algorithm. Gradient Descent algorithm is given below1.

Algorithm 1 Gradient Descent Algorithm

```
Input: a starting point($current\_x$) $x \in \mathbf{dom} f$ where f is a differentiable function, precision $\lambda$,
learning rate $\eta$ and let's assume $previous\_step\_size$ as $current\_x$ initially.
Output: some x hopefully minimizing f
repeat
    1. $previous\_x = current\_x$
    2. $current\_x \leftarrow current\_x - \eta * \nabla f(x)$
    3. $previous\_step\_size = |current\_x - previous\_x|$
until $previous\_step\_size > \lambda$
```

Your task is to implement the gradient descent algorithm where $f(x)$ can be derived substituting your registration number as follows.

Let us say your registration number is $E12e_1e_2e_3$. Then

$$f(x) = e_2 * x^{e_1 \bmod 5} - e_3 * x^{e_2 \bmod 5} - e_1 * x^{e_3 \bmod 5} + e_3$$

1. Plot the behavior of $f(x)$.

2. Briefly explains the selection of initial x value ($current\_x$).

3. What will happen when changing the learning rate (η) form small gradient size to large gradient size?

4. Comment on the result.

4 Submission

Submit a single .py file as [12|13]xxxlab02.py where xxx is your registration number. Add answers for questions 2,3 and 4 as comments in the same file.

5 Important

Make sure that you have the basic understanding of visualizations. This lab is really important for successive labs. If you do not understand any concepts, make sure you get some help from instructors.

6 Deadline

July 06, 23:59:59 GMT+5:30.