Introduction to python for AI

Alessia Mondolo

July 16, 2019

Academy AI

Introduction to MatplotLib

Matplotplib

Matplotlib is a plotting library that will allow us to make graphical representations of data. Matplotlib is a huge library but we will focus on pyplot.

Matplotlib comes installed with the conda enviroment. In other scenario we need to install it by means of pip, to install it we just need to run.

```
pip install matplotlib
```

The standard import of matplotlib is the following.

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

Jupyter Integration

Jupyter have a very nice integrations which lets you plot the graphs inline, to do so at the beginning of the notebook you must write `%matplotlib inline`

Figure

A figure is where each plot reside, we may do several plots inside the same figure. Plots by default will be painted in the last figure generated.

```
plt.figure()
```

Subplots can be generated inside the same figure by calling `add_subplot` over the object. This methods need the number of columns and rows and the position of the subfigure.

```
fig =plt.figure()
fig.add_subplot(2, 2, 1) # 2 rows, 2 cols, first element
fig.add_subplot(2, 2, 4) # 2 rows, 2 cols, last element
```

Axis, title and ticks I

When we create a plot we get an axis object, this object gives us the possibility of putting labels or changing the ticks of each axis (x, y, ...) and much more things.

```
fig =plt.figure()
ax =fig.add_subplot(1, 1, 1)
```

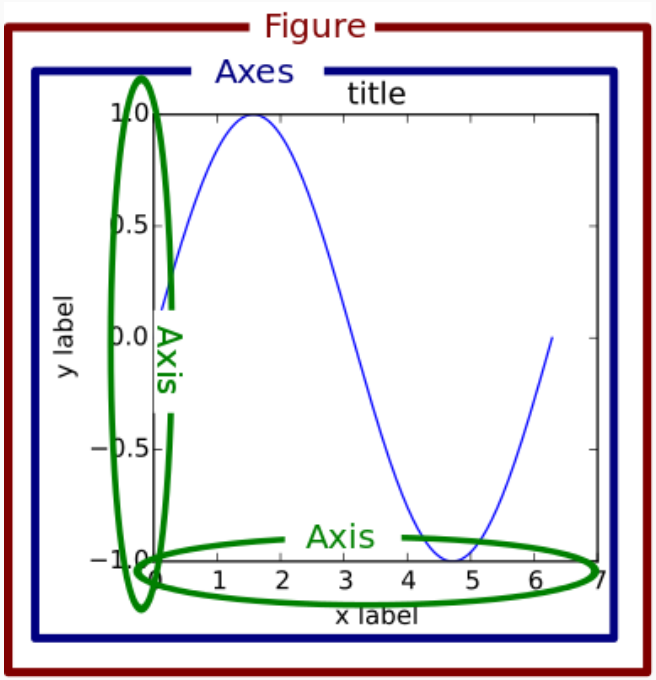
Ticks are the actual points that will be painted in the graph, each tick will have an associated label.

```
ax.set_xticks([-2, -1, 0, 1, 2])
ax.set_xticklabels(['-std2', '-std1', 'mean', 'std1', 'std2'])
```

Axis, title and ticks II

Some of the things that we can customize in the axis are the following.

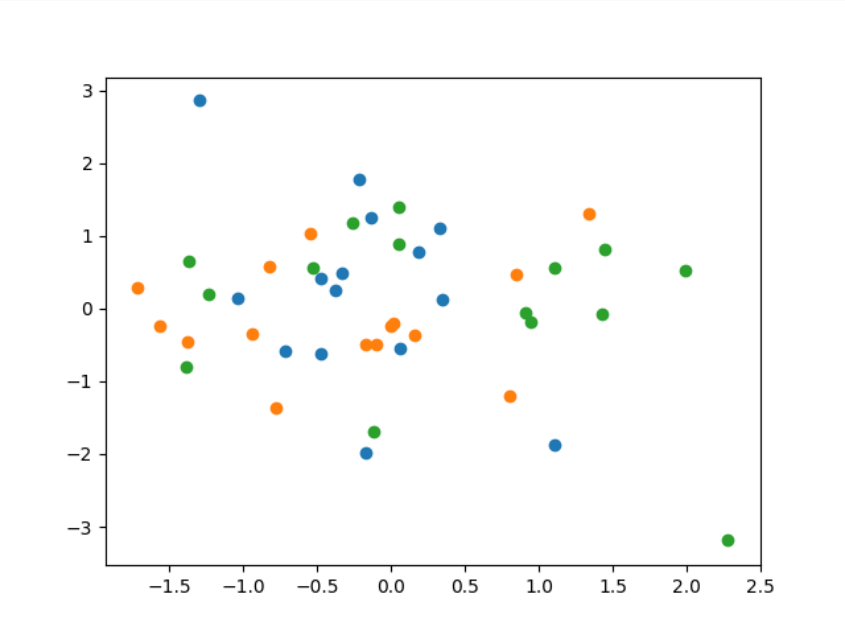
1. **xTicks** `a.set_xticks`
2. **xTicksLabels** `a.set_xtickslabels`
3. **yTicks** `a.set_yticks`
4. **yTicksLabels** `a.set_ytickslabels`
5. **Title** `a.set_title`



Scatter

A scatter graph is the one produced by single points, you need `x` and `y` coordinates in order to define a point, in matplotlib the method `plt.scatter` is the one used to perform this kind of graphs.

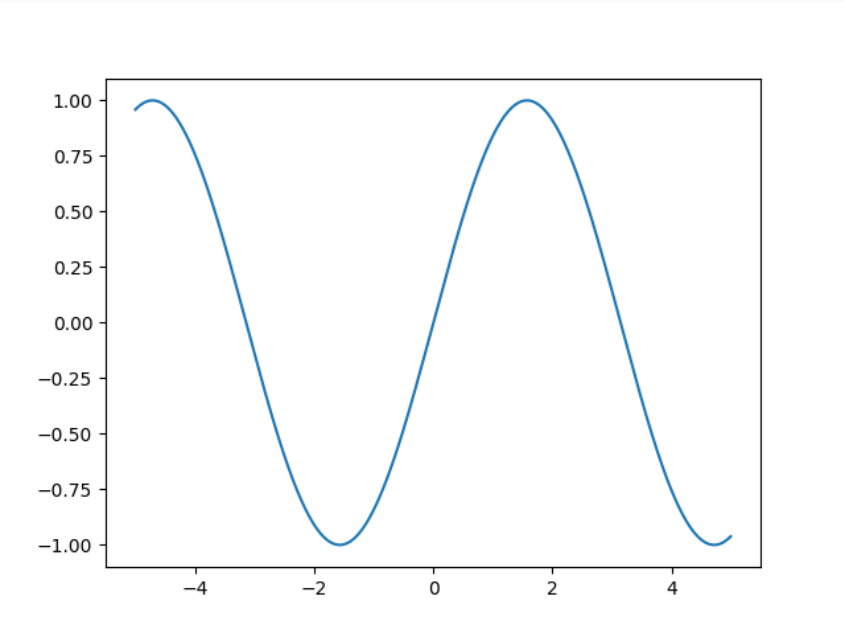
```
plt.scatter(
    np.random.normal(size=10), np.random.normal(size=10)
)
```



Line

Line plots are the ones produced by a function, you need to define x, y [and z]. The plotting is done with `plt.plot`.

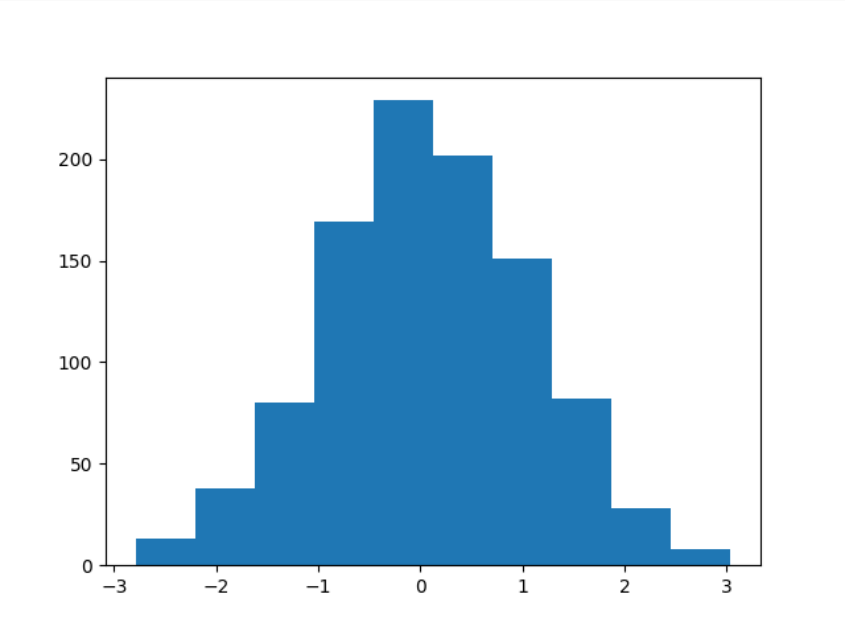
```
plt.plot(
    np.arange(-5, 6, 0.01), np.sin(np.arange(-5, 6, 0.01))
)
```



Histogram

A histogram is a graph representing the frequencies of data. Maplotlib only needs an array of vectors for calculating the histogram. The method `plt.hist` will produce this kind of graphs.

```
plt.hist(
    np.random.normal(size=100)
)
```



Pie chart

A pie chart is a special kind of chart that uses a circular representation. A vector is need by matplotlib to do this representation. Kepp in mind that values will be normalized. The method is `plt.pie`

```
plt.pie(
    [0.2, 0.3, 0.5]
)
```

The plot method

All these exposed methods have a common interface to define some kind of visual properties.

1. **legend** A text that will be displayed gibing name to the plots in the figure.
2. **style** In scatter and line plots you can define the type of line or point with a third parameter indicating color and type of point in a string more info here

Saving graphs

This graphs can be saved programatically in images by the method `plt.savefig`.

```
plt.savefig('figpath.png', dpi=400, bbox_inches='tight')
```

Pandas Integration

Pandas allows us to plot directly from a series by using the `plot` method together with the kind parameter. So for drawing a histogram for example we will do.

```
df =pd.Series([0.1, 0.2, 0.3, 0.5]).plot(kind='hist')
```

Introduction to Scikit-Learn interface

Matplotplib

Sklearn is the Swiss knife of machine learning, it comes with dozens of models out of the box and a huge community.

Sklearn comes installed with the conda enviroment. In other scenario we need to install it by means of pip, to install it we just need to run.

```
pip install sklearn
```

Models in sklearn are imported separately as for example.

```
from sklearn.ensemble import RandomForestClassifier
```

Types of models

Inside of sklearn we will find different types of models. I will just introduce the high level API of them.

- **Supervised models** Models to perform predictions.
- **Unsupervised models** Models to group data automatically.
- **Transformation models** Models to perform transformations in the data.

Supervised models

This kind of models are the most intuitive ones. You train them with data and expected outputs and later it will predict outputs for unseen data. To train the algorithm we call the `fit` method and to predict with it we call the `predict` function.

```
from sklearn.ensemble import RandomForestClassifier

clf =RandomForestClassifier().fit(X, y)
clf.predict(X)
```

Unupervised models

Other type of models, in this case it will not predict but find groups of similar elements inside data. To train the algorithm we call the `fit` method and to get the group of an unseen element we call the `predict` method.

```
from sklearn.cluster import KMeans

clf =KMeans().fit(X)
clf.predict(X)
```

Transformation models

This last kind of models transform our data. For example for dimensionality reduction tasks or normalization tasks. Their main method is `fit_tranform`

```
from sklearn.preprocessing import MinMaxScaler

transformed_data =MinMaxScaler().fit_transform(X)
```

Sklearn module structure

https://scikit-learn.org/stable/modules/classes.html

Sklearn cheat-sheet