クレジット:
UTokyo Online Educationデータマイニング入門 2018 森　純一郎

東京大学
THE UNIVERSITY OF TOKYO

CC BY NC ND

データマイニング入門

Pythonの基礎

Matplotlibライブラリ

Matplotlibライブラリにはグラフを可視化するためのモジュールが含まれています。以下では、Matplotlibライブラリのモジュールを使った、グラフの基本的な描画について説明します。

線グラフ

Matoplotlibライブラリを使用するには、まず `matplotlib` のモジュールをインポートします。ここでは、基本的なグラフを描画するための `matplotlib.pyplot` モジュールをインポートします。慣例として、同モジュールを `plt` と別名をつけてコードの中で使用します。

また、グラフで可視化するデータはリストや配列を用いることが多いため、 `numpy` モジュールも併せてインポートします。なお、 `%matplotlib inline` はjupyter notebook内でグラフを表示するために必要です。

`matplotlib` では、通常 `show()` 関数を呼ぶと描画を行いますが、 `inline` 表示指定の場合、 `show()` 関数を省略できます。この時、セルの最後に評価されたオブジェクトの出力表示を抑制するために、以下ではセルの最後の行にセミコロンをつけています。

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

以下では、 `pyplot` モジュールの **`plot ()`** 関数を用いて、リストの要素の数値をy軸の値としてグラフを描画しています。y軸の値に対応するx軸の値は、リストの各要素のインデックスとなっています。

In [ ]:
```
# plotするデータ
d =[0, 1, 4, 9, 16]

# plot関数で描画
plt.plot(d);
```

plot() 関数では、x, yの両方の軸の値を引数に渡すこともできます。

In [ ]:
```
# plotするデータ
x =[0, 1, 2, 3, 4]
y =[0, 1, 2, 3, 4]

# plot関数で描画
plt.plot(x,y);
```

以下のようにグラフを複数まとめて表示することもできます。複数のグラフを表示すると、線ごとに異なる色が自動で割り当てられます。 plot() 関数ではグラフの線の色、形状、データポイントのマーカの種類を、それぞれ以下のように linestyle , color , marker 引数で指定して変更することができます。それぞれの引数で指定可能な値は以下を参照してください。

- linestyle (https://matplotlib.org/api/_as_gen/matplotlib.lines.Line2D.html#matplotlib.lines.Line2D.set
- color (https://matplotlib.org/api/colors_api.html?highlight=color#module-matplotlib.colors)
- marker (https://matplotlib.org/api/markers_api.html#module-matplotlib.markers)

plot() 関数の label 引数にグラフの各線の凡例を文字列として渡し、 **legend** () 関数を呼ぶことで、グラフ内に凡例を表示できます。 legend() 関数の loc 引数で凡例を表示する位置を指定することができます。引数で指定可能な値は以下を参照してください。

- lengend()関数 (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.legend.html#matplotlib.pyplot.legend)

In [ ]:
```
# plotするデータ
data =[0, 1, 4, 9, 16]
x =[0, 1, 2, 3, 4]
y =[0, 1, 2, 3, 4]

# plot関数で描画。線の形状、色、データポイントのマーカ、凡例を指定
plt.plot(x,y, linestyle='--', color='blue', marker='o', label="linear"
plt.plot(data, linestyle=':', color='green', marker='*', label="quad")
plt.legend();
```

pyplot モジュールでは、以下のようにグラフのタイトルと各軸のラベルを指定して表示することができます。タイトル、x軸のラベル、y軸のラベル、はそれぞれ **title** () 関数、 **xlabel** () 関数、 **ylabel** () 関数に文字列を渡して指定します。また、 **grid** () 関数を用いるとグリッドを併せて表示することもできます。グリッドを表示させたい場合は、 grid() 関数に True を渡してください。

In [ ]:
```
# plotするデータ
data =[0, 1, 4, 9, 16]
x =[0, 1, 2, 3, 4]
y =[0, 1, 2, 3, 4]

# plot関数で描画。線の形状、色、データポイントのマーカ、凡例を指定
plt.plot(x,y, linestyle='--', color='blue', marker='o', label="linear")
plt.plot(data, linestyle=':', color='green', marker='*', label="quad")
plt.legend()

plt.title("My First Graph") # グラフのタイトル
plt.xlabel("x") #x軸のラベル
plt.ylabel("y") #y軸のラベル
plt.grid(True); #グリッドの表示
```

グラフを描画するときのプロット数を増やすことで任意の曲線のグラフを作成することもできます。以下では、 numpy モジュールの arange() 関数を用いて、$-\pi$からπの範囲を0.1刻みでx軸の値を配列として準備しています。そのx軸の値に対して、 numpy モジュールの cos() 関数と sin() 関数を用いて、y軸の値をそれぞれ準備し、 cos カーブと sin カーブを描画しています。

In [ ]:
```
#  グラフのx軸の値となる配列
x = np.arange(-np.pi, np.pi, 0.1)

# 上記配列をcos, sin関数に渡し, y軸の値として描画
plt.plot(x,np.cos(x))
plt.plot(x,np.sin(x))

plt.title("cos ans sin Curves")  # グラフのタイトル
plt.xlabel("x") #x軸のラベル
plt.ylabel("y") #y軸のラベル
plt.grid(True); #グリッドの表示
```

プロットの数を少なくすると、曲線は直線をつなぎ合わせることで描画されるていることがわかります。

In [ ]:
```
x = np.arange(-np.pi, np.pi, 0.5)
plt.plot(x,np.cos(x), marker='o')
plt.plot(x,np.sin(x), marker='o')
plt.title("cos ans sin Curves")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True);
```

散布図

散布図は、 pyplot モジュールの **scatter ()** 関数を用いて描画できます。以下では、ランダムに生成した20個の要素からなる配列x,yの各要素の値の組みを点としてプロットした散布図を表示しています。プロットする点のマーカーの色や形状は、線グラフの時と同様に、 color , marker 引数で指定して変更することができます。加えて、 s , alpha 引数で、それぞれマーカーの大きさと透明度を指定することができます。

In [ ]:
```
#  グラフのx軸の値となる配列
x = np.random.rand(20)
#  グラフのy軸の値となる配列
y = np.random.rand(20)

# scatter関数で散布図を描画
plt.scatter(x, y, s=100, alpha=0.5);
```

以下のように、 plot() 関数を用いても同様の散布図を表示することができます。この時、 plot() 関数では、プロットする点のマーカーの形状を引数に指定しています。

In [ ]:
```
x = np.random.rand(20)
y = np.random.rand(20)
plt.plot(x, y, 'o', color='blue');
```

棒グラフ

棒グラフは、 pyplot モジュールの **bar ()** 関数を用いて描画できます。以下では、ランダムに生成した10個の要素からなる配列 y の各要素の値を縦の棒グラフで表示しています。 x は、x軸上で棒グラフのバーの並ぶ位置を示しています。ここでは、 numpy モジュールの arange() 関数を用いて、1から10の範囲を1刻みでx軸上のバーの並ぶ位置として配列を準備しています。

In [ ]:
```
# x軸上で棒の並ぶ位置となる配列
x = np.arange(1, 11, 1)
#  グラフのy軸の値となる配列
y = np.random.rand(10)

# bar関数で棒グラフを描画
plt.bar(x,y);
```

ヒストグラム

ヒストグラムは、 pyplot モジュールの **hist** () 関数を用いて描画できます。以下では、 numpy モジュールの random.randn() 関数を用いて、正規分布に基づく1000個の数値の要素からなる配列を用意し、ヒストグラムとして表示しています。 hist() 関数の bins 引数でヒストグラムの箱（ビン）の数を指定します。

In [ ]:
```
# 正規分布に基づく1000個の数値の要素からなる配列
d = np.random.randn(1000)

# hist関数でヒストグラムを描画
plt.hist(d, bins=20);
```

ヒートマップ

impshow() 関数を用いると、以下のように行列の要素の値に応じて色の濃淡を変えることで、行列をヒートマップとして可視化することができます。 colorbar() 関数は行列の値と色の濃淡の対応を表示します。

In [ ]:
```
# 10行10列のランダム要素からなる行列
a = np.random.rand(100).reshape(10,10)

# imshow関数でヒートマップを描画
im=plt.imshow(a)
plt.colorbar(im);
```

グラフの画像ファイル出力

savefig () 関数を用いると、以下のように作成したグラフを画像としてファイルに保存することができます。

In [ ]:
```
x = np.arange(-np.pi, np.pi, 0.1)
plt.plot(x,np.cos(x), label='cos')
plt.plot(x,np.sin(x), label='sin')
plt.legend()
plt.title("cos ans sin Curves")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)

# savefig関数でグラフを画像保存
plt.savefig('cos_sin.png');
```