EML4930/EML6934: Lecture 06

More advanced topics in matplotlib

Charles Jekel

October 5, 2017

Review syllabus

Any questions with Matplotlib pyplot?

Topics for today

- Histograms
- Contour plots
- 3D scatter plot
- 3D line plot
- 3D surface plot
- 2D Line through high dimensional space

What is a histogram?

Histogram:

It's a type of bar graph used to approximate the probability distribution function of a continuous variable. You divide a sampled variable into a number of *bins*, and then count the number of sample occurrences in each bin. You'll see the bins as the bar widths, and the frequency as the bar heights.

`https://en.wikipedia.org/wiki/Histogram`

A number of distributions in np.random to sample from

https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.random.html

Let's draw samples from the Gumbel distribution

```
import numpy as np
mu = 4.0
beta = 0.2
# let's draw 1000 random samples from the Gumbel distribution
samples = np.random.gumbel(mu, beta, 1000)
```

Let's create a histogram plot of the samples

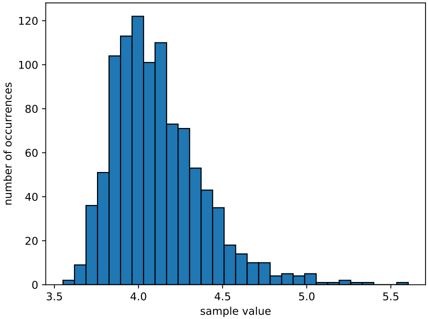
Take a look at the documentation:
https://matplotlib.org/api/pyplot_api.html?highlight=matplotlib%20pyplot%20hist#matplotlib.pyplot.hist

```
matplotlib.pyplot.hist(x, bins=None, range=None, normed=False,
 weights=None, cumulative=False, bottom=None, histtype='bar',
 align='mid', orientation='vertical', rwidth=None, log=False,
 color=None, label=None, stacked=False, hold=None,
 data=None, **kwargs)
```

There is a lot of options here. I'll try to mention the important ones.

Plotting a histogram of the Samples

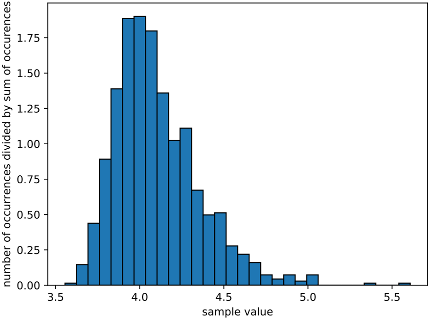
```
from matplotlib.pyplot as plt
plt.figure()
plt.hist(samples, bins=30, edgecolor='black')
plt.xlabel('sample value')
plt.ylabel('number of occurrences')
plt.show()
```



You generally want to work with a normalized histogram

The integral of a probability distribution function (PDF) will be 1 by definition. The normed flag divides the frequency of each bin by the total number of samples.

```
plt.hist(samples, bins=30, edgecolor='black', normed=True)
```



Note: plot created from a different random sample.

Summary of plt.hist

- Histograms are sensitive to the number of bins
- Various options to control range, bin size...
- Use normed=True for PDFs
- If you don't need the plot, and would work with the bins and values themselves use

  `np.histogram(a, bins=10,rane=None,Normed=False,...)`

Creating contour plots

It's useful to represent 3D surface in a 2D plot. Generally we create contour plots to do so.

```
# plot contour lines of a Function F(x,y)
plt.contour(x,y,F)

# create filled contour plot of a Function F(x,y)
plt.contourf(x,y,F)
```

Read the documentation and examples: `https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.contour`
`https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.contourf`

Let's consider the Rosenbrock function

$$f(x, y) = (1-x)^2 + 100(y-x^2)^2$$
$$-1 \leq y \leq 3$$
$$-2 \leq x \leq 2$$

```
x = np.linspace(-2,2,100)
y = np.linspace(-1,3,100)
# we use np.meshgrid to create an x,y grid
x,y = np.meshgrid(x,y)
f = (1.0-x)**2 + (100.0*((y-(x**2))**2))
```

Basic contour plots

In this case we are telling the function to automatically create 100 levels

```
# just the lines
plt.figure()
plt.contour(x,y,f,100,
    cmap=plt.cm.viridis)
plt.colorbar()
plt.show()

# filled contour plot
plt.figure()
plt.contourf(x,y,f,100,
    cmap=plt.cm.viridis)
plt.colorbar()
plt.show()
```

There are numerous types of color maps

Check them all out at
https://matplotlib.org/users/colormaps.html

My favorite are viridis (now default), plasma, inferno, and magma.

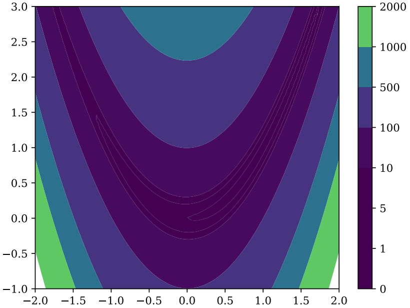
```
plt.figure()
plt.contourf(x,y,f,100,
    cmap=plt.cm.plasma)
plt.colorbar()
plt.show()

plt.figure()
plt.contourf(x,y,f,100,
    cmap=plt.cm.hot)
plt.colorbar()
plt.show()
```

You can manually specify the levels as a list

```
plt.figure()
levels=[0, 1, 5, 10, 100, 500, 1000, 2000]
plt.contourf(x,y,f,levels)
plt.colorbar()
plt.show()
```

Contours - summary

There are many many more features of contours. My suggestion is to read the documentation if you need to do something more advance with contours.



The documentation includes a bunch of useful examples: `https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.contour`

Intro to the object orient plotting interface

Last class we did:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0.0, 2.0*np.pi, 25)
plt.figure()
plt.plot(x,np.cos(x), '--o', label='cos')
plt.plot(x,np.sin(x), '-.s', label='sin')
plt.grid(True)
plt.legend()
plt.xlabel('x axis')
plt.ylabel('y axis')

plt.show()
```

Same result - but with OOP

```python
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0.0, 2.0*np.pi, 25)

# the figure and axes of the plot are now objects
fig, ax = plt.subplots()

# you plot on the axes
ax.plot(x,np.cos(x), '--o', label='cos')
ax.plot(x,np.sin(x), '-.s', label='sin')
ax.grid(True)
ax.legend()
# However these labels are different!
ax.set_xlabel('x axis')
ax.set_ylabel('y axis')

fig.show()
```

Alternative OOP syntax with ax.set()

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0.0, 2.0*np.pi, 25)
fig, ax = plt.subplots()
ax.plot(x,np.cos(x), '--o', label='cos')
ax.plot(x,np.sin(x), '-.s', label='sin')
ax.grid(True)
ax.legend()

# using set to set multiple items
ax.set(xlim=(-1,7), ylim=(-2,2),
    xlabel='x', ylabel='y',
    title='cos and sin plot')

fig.show()
```

Reasons to use OOP Matplotlib interface

For the simple style of plots (like the ones covered thus far), I use the script-based interface.

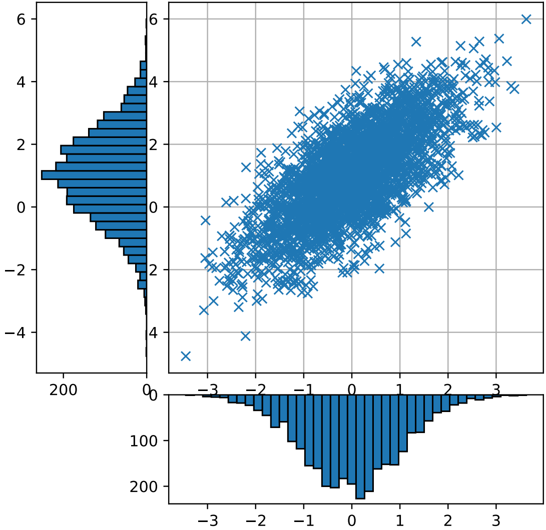
For the complex plots such as

- subplots
- 3D plots
- shapes and patches

I'll use the OOP interface.

There are basically things you can do with the OOP interface that you can't in the interface.

For instance something like this



Code to generate 1 of 2

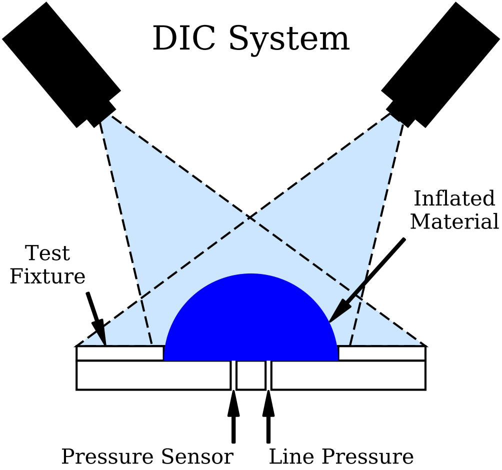
```
# Create some normally distributed data
mean = [0, 1]
cov = [[1, 1], [1, 2]]
x, y = np.random.multivariate_normal(mean, cov, 3000).T
# Set up the axes with gridspec
fig = plt.figure(figsize=(6, 6))
grid = plt.GridSpec(4, 4, hspace=0.2, wspace=0.2)
main_ax = fig.add_subplot(grid[:-1, 1:])
y_hist = fig.add_subplot(grid[:-1, 0], sharey=main_ax)
x_hist = fig.add_subplot(grid[-1, 1:], sharex=main_ax)
```

Code to generate 2 of 2

```
# scatter points on the main axes
main_ax.plot(x, y, 'x')
main_ax.grid(True)
# histogram on the attached axes
x_hist.hist(x, 40, orientation='vertical',
 edgecolor='black')
x_hist.invert_yaxis()
y_hist.hist(y, 40, orientation='horizontal',
 edgecolor='black')
y_hist.invert_xaxis()
fig.show()
```

Made with Matplotlib...

Source code uploaded: Lectures/06/shapeDIC.py



3D scatter plot in Matplotlib

Alternative demo: https://matplotlib.org/examples/mplot3d/scatter3d_demo.html

```
# we have a new import!!!
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x,y, f, 'ob')
ax.set( xlabel='X Label',
 ylabel ='Y Label',
 zlabel='Z Label')
fig.show()
```

3D line plot in Matplotlib

Alternative demo:

https://matplotlib.org/examples/mplot3d/lines3d_demo.html

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# let's randomly choose some lines to plot in the domain
ax.plot(x[1],y[1], f[1], '-')
ax.plot(x[9],y[9], f[9], '-')
ax.plot(x[35],y[35], f[35], '-')
ax.scatter(x,y, f, 'ob')
ax.set( xlabel='X Label',
 ylabel ='Y Label',
 zlabel='Z Label')
fig.show()
```

3D surface plot in Matplotlib

Alternative demo:

https://matplotlib.org/examples/mplot3d/lines3d_demo.html

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# let's plot the surface
# please note this only works if your x,y,z points on on a grid
# ie created with np.meshgrid !
ax.plot_surface(x,y, f)
ax.set( xlabel='X Label',
 ylabel ='Y Label',
  zlabel='Z Label')
fig.show()
```

You can do a bunch more with Matplotlib and 3D

Please check out the 3D examples

`https://matplotlib.org/mpl_toolkits/mplot3d/tutorial.html`

This concludes the basic/advance topics of Matplotlib.

High dimensional visualization - line between two points

Consider two data points

$$\boldsymbol{a} = [x_1, x_2, \cdots, x_n] \tag{1}$$
$$\boldsymbol{b} = [x_1, x_2, \cdots, x_n] \tag{2}$$
$$\tag{3}$$

and a high dimensional function $f(x_1, x_2, \cdots, x_n)$. We can reduce this high dimensional space to a one dimensional line between the two points $\boldsymbol{a}$ and $\boldsymbol{b}$.

Consider the vector $\boldsymbol{u}$ from $\boldsymbol{a}$ to $\boldsymbol{b}$

$$\boldsymbol{u} = \boldsymbol{b} - \boldsymbol{a} \tag{4}$$

where $\boldsymbol{u}$ indicates the direction from the point $\boldsymbol{a}$ to the point $\boldsymbol{b}$

High dimensional visualization - line between two points

We can reduce the n dimensional problem into a parametric line between points $\boldsymbol{a}$ and point $\boldsymbol{b}$. The variable that will dictate the step along the line will be $\boldsymbol{r}$ such that the new high dimensional problem is represented as

$$\boldsymbol{r} = \boldsymbol{a} + \lambda \boldsymbol{u} \tag{5}$$

$$0 \leq \lambda \leq 1 \tag{6}$$

Remember that $\boldsymbol{r} = [x_1, x_2, \cdots, x_n]$ for some λ. To evaluate the line between $\boldsymbol{a}$ and $\boldsymbol{b}$, we evaluate

$$f(\boldsymbol{r}) \tag{7}$$

as we vary λ between 0 and 1. When $\lambda = 0$ we'll be at point $\boldsymbol{a}$. When $\lambda = 1$ we'll be at point $\boldsymbol{b}$.

Example: Rosenbrock function

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2 \tag{8}$$

where

$$\boldsymbol{a} = [0.33, 0.57] \tag{9}$$

$$\boldsymbol{b} = [-0.11, -0.2] \tag{10}$$

Problem: Visualize the line between $\boldsymbol{a}$ and $\boldsymbol{b}$.

Rosenbrock function visualization - between two points

```
import numpy as np
import matplotlib.pyplot as plt
a = np.array([0.33, .57])
b = np.array([-0.11, -0.2])
u = b-a #
lam = np.linspace(0,1,100)
x = np.zeros(100)
y = np.zeros(100)
for i,j in enumerate(lam):
    r = (a + (j*u))
    x[i] = r[0]
    y[i] = r[1]
F = (1.0-x)**2 + (100.0*((y-(x**2))**2))
plt.figure()
plt.plot(lam,F)
plt.ylabel('F'); plt.xlabel(r'$\lambda$')
plt.show()
```

Result