EML4930/EML6934: Lecture 05

More NumPy and some matplotlib

Charles Jekel

September 28, 2017

Reminder Quiz at the end of this class

Issues with HW?

What am I going to cover this lecture

- review arrays
- difference between array dimensions
- array shaping
- saving and loading numpy arrays
- matplotlib

Let's consider these two arrays

```
import numpy as np
a = np.array([6, 9, 2, 3, 6])
b = np.array([[6], [9], [2], [3], [6]])

# what are the dimensions
print(a.ndim)
print(b.ndim)

# what are the shapes
print(a.shape)
print(b.shape)

# let's print the transpose of a
print(a)
print(a.T)
```

Let's consider these two arrays

```
# let's print the transpose of b
c = b.T
print(b)
print(c)

# what has change?
print(b.shape)
print(c.shape)

# what is the difference between the following?
print(a*a)
print(np.dot(a,a))
```

Saving NumPy arrays

```
np.save(file, arr)
```

Save an array to a binary file in NumPy .npy format.

```
np.save('a_vect',a)
```

This creates a_vect.npy file of the vector a.

Loading NumPy arrays

```
np.load(file)
```

Load arrays or pickled objects from .npy, .npz or pickled files.

```
a_vect = np.load('a_vect.npy')
```

This loads the a_vect.npy binary file into the a_vect vector.

Saving NumPy arrays as plain text and loading the file

```
np.savetxt(fname,X)
```

Save the array X to a text file fname.

```
a = numpy.loadtxt(fname)
```

Load data from a text file named fname and store this as array a.

Each row in the text file must have the same number of values.

NumPy reshape

```
a = np.arange(6)
b = a.reshape((3,2))
c = a.reshape((2,3))
d = a.reshape((6,1))
print(b)
print(c)
print(d)
```

```
out:
[[0 1]
 [2 3]
 [4 5]]
[[0 1 2]
 [3 4 5]]
[[0 1 2 3 4 5]]
```

NumPy array concatenation - or joining

```
x = np.array([4, 5, 0, 3, 7])
y = np.array([3, 4, 9, 7, 5])
z = np.concatenate([x,y])
print(z)

# you can you use np.vstack to vertically stack arrays
w = np.vstack([x,y])

# you can use np.hstack to horizontally stack arrays
v = np.hstack([w,w])
```

flatten any ndarray into one dimension

```
k = np.random.random((5,3,2,6,8))
print(k.ndim)
k_flat = k.flatten()
print(k_flat.ndim)
print(k_flat.size)

out:
5
1
1440
```

matplotlib

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For a sampling, see the screenshots, thumbnail gallery, and examples directory

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

`https://matplotlib.org/`

For reference

Dr. Jake VanderPlass Python Data Science Handbook: Essential Tools for Working with Data.

Chapter 4: Visualization with Matplotlib

`http://nbviewer.jupyter.org/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/Index.ipynb`

matplotlib pyplot - MATLAB like plotting framework

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0.0, 2.0*np.pi, 25)

# you can specify a figure number or string name
# but by default plt.figure() will create a number ordered
# figure name Ex: plt.figure('my figure') or plt.figure(1)
plt.figure()
# create a line plot by default
plt.plot(x,np.cos(x))
# show the plot
plt.show()
```

matplotlib linestyles

Linestyle	Description
'-' or 'solid'	solid line
'–' or 'dashed'	dashed line
'-.' or 'dashdot'	dash-dotted line
':' or 'dotted'	dotted line
'None'	draw nothing
' '	draw nothing
''	draw nothing

Plotting cosine with a dashed line

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0.0, 2.0*np.pi, 25)

plt.figure()

# all you need to do is pass the linestyle as an attribute
plt.plot(x,np.cos(x), '--')
plt.show()
```

Scatter plot markers

and more at - `https://matplotlib.org/api/markers_api.html`

Marker	Description
"."	point
","	pixel
"o"	circle
"v"	triangle_down
"<"	triangle_left
">"	triangle_right
"1"	tri_down
"2"	tri_up
"3"	tri_left
"4"	tri_right
"8"	octagon
"s"	square
"p"	pentagon
"P"	plus (filled)

Scatter plot cosine

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0.0, 2.0*np.pi, 25)

plt.figure()

# all you need to do is pass the marker into plot
plt.plot(x,np.cos(x), 'o')
plt.show()
```

You can easily combine markers and linestyles

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0.0, 2.0*np.pi, 25)

plt.figure()

# this will plot a -- linestyle with circles at the data points
plt.plot(x,np.cos(x), '--o')
plt.show()
```

basic built-in colors in matplotlib

and more at - `https://matplotlib.org/api/colors_api.html`

Code	Color
b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

You can specify the built in color into plot

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0.0, 2.0*np.pi, 25)

plt.figure()

# forcing the line and dot color to be blue
plt.plot(x,np.cos(x), 'b--o')
plt.show()
```

Alternatively use color=”

```
# specify color by name
plt.plot(x, np.cos(x), color='blue')

# short color code (rgbcmyk)
plt.plot(x, np.cos(x), color='g')

# Grayscale between 0 and 1
plt.plot(x, np.cos(x), color='0.75')

# Hex code (RRGGBB from 00 to FF)
plt.plot(x, np.cos(x), color='#FFDD44')

# RGB tuple, values 0 and 1
plt.plot(x, np.cos(x), color=(1.0,0.2,0.3))

# all HTML color names supported
plt.plot(x, np.cos(x), color='chartreuse')
```

Adjusting the plot with axes limit

```
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), 'b--o')

# set the x axis limit
plt.xlim(-1,7)

# set the y axis limit
plt.ylim(-2,2)

plt.show()
```

Adding a grid

https://matplotlib.org/api/pyplot_api.html?highlight=matplotlib%20pyplot%20grid#matplotlib.pyplot.grid

grid(b=None, which='major', axis='both', **kwargs)

kwargs are used to set the grid line properties, e.g.,:

```
ax.grid(color='r', linestyle='-', linewidth=2)
```

```
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), 'b--o')
# create a grid
plt.grid(True)
plt.show()
```

Labels and legend

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

# add label='my_label'
plt.plot(x,np.cos(x), '--o', label='cos')
plt.plot(x,np.sin(x), '-.s', label='sin')
plt.grid(True)

# add legend
plt.legend()
# legend automatically chooses the location
# but you can specify the simple quadrant based location as
# plt.legend(loc=1) puts the legend in the first quadrant

plt.show()
```

matplotlib title and axis label

```
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()
# adding a title
plt.title('Cos and sin')

# x and y axis labels
plt.xlabel('x axis')
plt.ylabel('y axis')

plt.show()
```

matplotlib pyplot savefig

https://matplotlib.org/api/pyplot_api.html?highlight=matplotlib%20pyplot%20savefig#matplotlib.pyplot.savefig

```
plt.savefig(fname, dpi=None, facecolor='w', edgecolor='w',
        orientation='portrait', papertype=None, format=None,
        transparent=False, bbox_inches=None, pad_inches=0.1,
        frameon=None)
```

How I normally create publication quality pictures:

```
plt.savefig('my_fig.pdf', dpi=600, format='pdf',
            bbox_inches='tight')
```

Most backends support png, pdf, ps, eps and svg.

saving my cosine and sin plot

Depending on your active Python interpreter, you'll have issues with plt.show()

```
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.savefig('my_fig.pdf', dpi=600, format='pdf',
            bbox_inches='tight')
```

Created this plot