Math with Python

Programming Lecture 2

Nicholas Dwork

Numpy

A Python package that provides numerical routines
- **Vectors**
- **Matrices**
- **Matrix Methods**

You must import Numpy before you use it

```
import numpy as np
# Can then use numpy packages with np later
```

Arrays

An array is an ordered set of numbers

```
import numpy as np
array1 = np.array( [2 1 4] )  # makes array with values
array2 = np.ndarray( [5,4] )   # makes array of size 5x4

array3 = np.array( [ [ 2 1 4 ], [ 8 9 8 ] ] )
   # Makes two dimensional array with values

array4 = np.zeros(  [5, 4] )  # makes array of all zeros
array4[1,1] = 8  # puts an 8 in the 1,1 position
```

Arrays

An array is an ordered set of numbers

```
import numpy as np
array5 = np.linspace(0, np.pi, 100 )
   # array of 100 points evenly spaced between 0 and pi
```

Size of Array

The shape **method provides the size of the array or matrix**

Suppose A **is a matrix**

A.shape[0] is the number of rows

A.shape[1] is the number of columns

Matrix Multiplication

Matrix-Vector multiplication

Multiply matrix A with vector x

out = A.dot(x)

Matrix-Matrix multiplication

Multiply matrix A with matrix B

out = A.dot(B)

Solving Linear Systems

Suppose you want x such that A x = b

```
import numpy as np
tmp = np.linalg.lstsq( A, b )
xHat = tmp[0]

# to get the Pseudo-inverse of A
A_dagger = np.linalg.pinv( A )
```

Compute the Norm

```
import numpy as np
normX = np.linalg.norm( x )
```

Plotting

Plotting is encapsulated in the matplotlib package
We will use the pyplot subpackage

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

x = np.linspace(0, np.pi, 100 )
   # array of 100 points evenly spaced between 0 and pi
y = np.sin(x)
plt.plot(x,y)
plt.show()
```

Save and Load Data

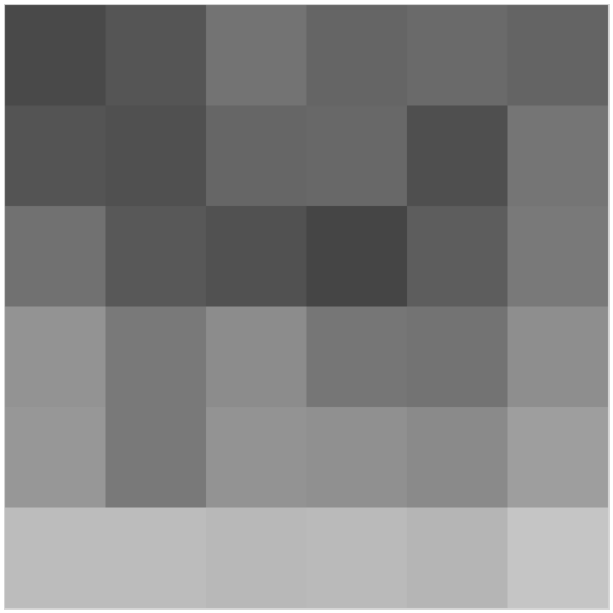
Also part of numpy

```
import numpy as np

a = np.linspace( 0, 1, 1000 )
np.save( 'aFile.npy', a )    # save data as binary file
np.savetxt( 'aFile.txt', a )  # save data as text file

np.load( 'aFile.npy', loadedBinaryA )
np.load( 'aFile.txt', loadedTxtA )
```

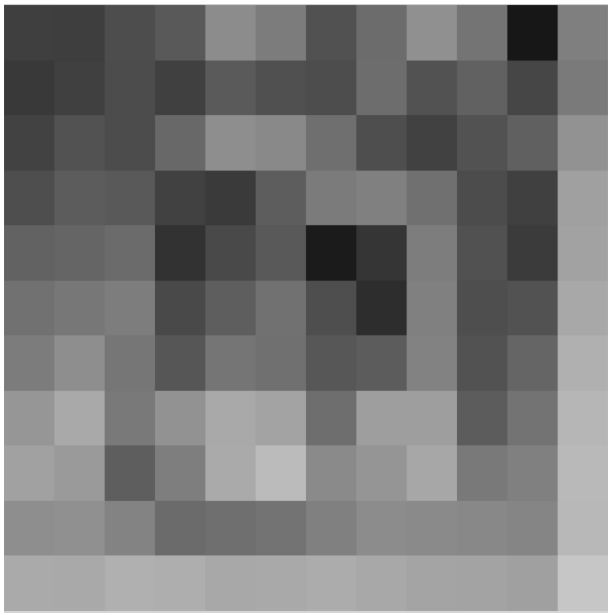
2D Array

57	67	96	82	87	81
66	63	83	85	62	98
94	70	64	53	75	102
129	102	121	99	96	123
133	102	129	125	119	140
174	174	170	172	166	184

Rather than showing the numbers, we can show corresponding colors. 0=black, and 255=white.



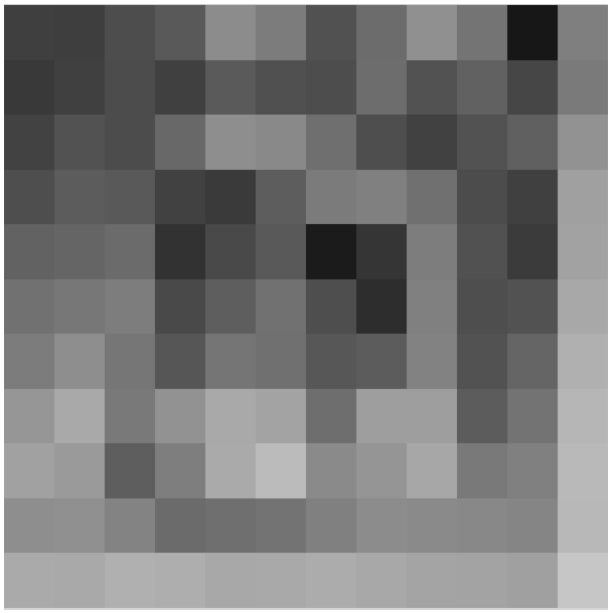
Here's a larger array.
0=black, and 255=white.



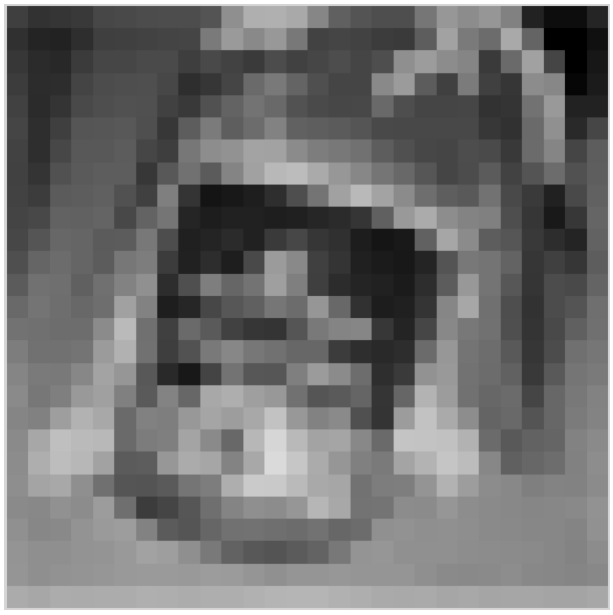
We can always go back to the array of numbers.

48	47	60	71	121	105	64	89	125	97	18	108
43	49	59	49	72	63	60	90	65	78	54	103
51	65	59	85	123	118	92	61	50	65	77	128
61	74	71	50	44	75	104	109	93	59	49	143
79	82	88	38	57	71	21	40	106	64	45	145
94	100	106	57	76	94	61	34	109	61	65	152
105	123	99	68	98	93	69	74	111	65	82	160
132	153	102	128	153	146	91	141	141	74	96	167
144	136	76	107	154	173	119	131	150	102	109	171
123	125	112	88	92	96	109	121	119	117	114	170
154	153	160	158	152	153	156	152	147	146	143	186

Here's a larger array.
0=black, and 255=white.

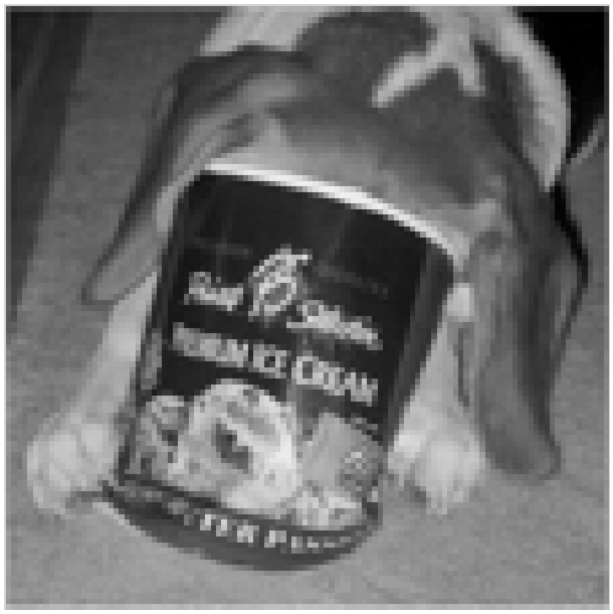


Here's an even larger array. Now we have too many numbers to display on this screen.
0=black, and 255=white.



And larger …
0=black, and 255=white.

And larger
0=black, and 255=white.



Still larger. At this point, our eye can no longer discern most of the individual pixels.
0=black, and 255=white.

Largest.
0=black, and 255=white.

Images

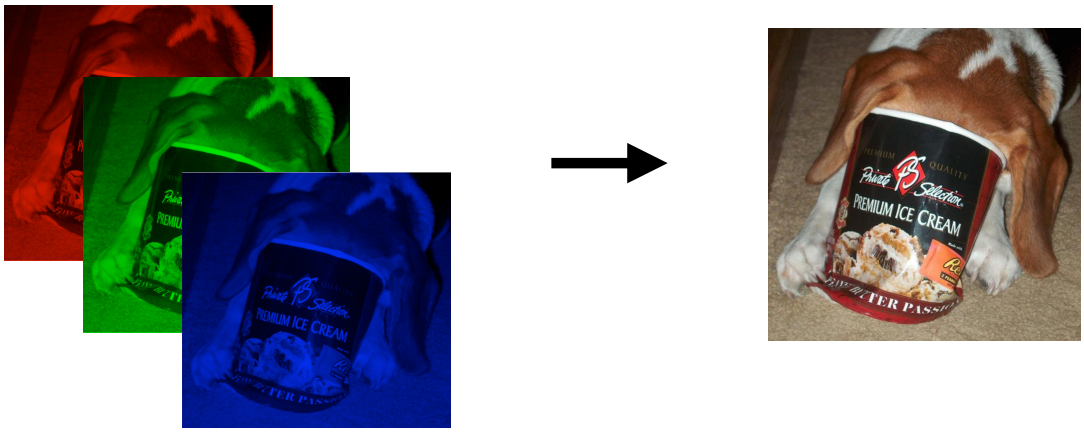
Big Conclusion: Images are just 2D arrays that are displayed in an interesting way!

At some point, your eye can no longer distinguish the individual pixels.

In Python, images and 2D arrays are exactly the same.

Color Image

A color image is three different arrays.
The computer displays one of the arrays for red, one for green, and one for blue.



Save and Load Images

Part of the image subpackage of matplotlib

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

imgArray = img.imread( 'imgFile.png' )
imgArray = imgArray[1:50,1:50]
img.imsave( 'imgFile.png', imgArray )
```