BioE 145 Assignment 1 Due Tuesday 1/28 11:59 PM

Note: Turn in your submission to this assignment in Gradescope by Tuesday January 28th, 11:59 PM. Attach a PDF printout of your completed IPython notebook from lab as an appendix, as well as any code used to find your answers to the following questions.

Part 1: Linear Algebra Review

1. Rank and Eigenvectors

   (a) Determine the eigenvalues and eigenvectors for the following matrices.

$$\begin{bmatrix} 1 & 6 \\ 4 & 6 \end{bmatrix}, \begin{bmatrix} 1 & 1.5 \\ 4 & 6 \end{bmatrix} \tag{1}$$

   For the first matrix:

$$\lambda_1 = -2, \quad v_1 = \begin{bmatrix} -0.894 \\ 0.447 \end{bmatrix} \qquad \lambda_2 = 9, \quad v_2 = \begin{bmatrix} -0.6 \\ -0.8 \end{bmatrix} \tag{2}$$

   For the second matrix:

$$\lambda_1 = 0, \quad v_1 = \begin{bmatrix} -0.832 \\ 0.555 \end{bmatrix} \qquad \lambda_2 = 7, \quad v_2 = \begin{bmatrix} -0.242 \\ -0.970 \end{bmatrix} \tag{3}$$

   (b) For both of the above, determine the rank and dimension of the null space.

   First matrix: rank = 2, dim(N) = 0
   Second matrix: rank = 1, dim(N) = 1

2. Show that a symmetric matrix A, with dimension n by n, can be expressed as

$$A = \sum_{i=1}^{n} \lambda_i v_i v_i^T \tag{4}$$

   where v_i is a vector in $\mathbb{R}^n$ and λ_i is a scalar in $\mathbb{R}$. What do they represent in relation to A?

   Hint: Use the following expansion for an n by n diagonal matrix D.

$$D = \sum_{i=1}^{n} D_{i,i} e_i e_i^T \tag{5}$$

   where e_i is the i^{th} vector in the standard basis of $\mathbb{R}^n$.

A symmetric matrix can be written as

$$A = QDQ^T \tag{6}$$

$$A = Q\left(\sum_{i=1}^{n} D_{i,i} e_i e_i^T\right) Q^T \tag{7}$$

$$A = \sum_{i=1}^{n} D_{i,i} Q e_i e_i^T Q^T \tag{8}$$

$$A = \sum_{i=1}^{n} D_{i,i} Q_i Q_i^T \tag{9}$$

$$A = \sum_{i=1}^{n} \lambda_i v_i v_i^T \tag{10}$$

where Q_i is the i^{th} column of Q.

λ_i is an eigenvalue of A and v_i is the corresponding eigenvector of A.

BioE 145 Assignment 1 Due Tuesday 1/28 11:59 PM

Part 2: Probability Review

1. You are analyzing the nucleotide composition of an animal genome and you notice a modest AT bias. The genome is 17/32 AT (53.125%) and only 15/32 GC (46.875%). You then start to look at the frequencies of dinucleotides.

   (a) If nucleotides are independent, what is the probability of 5'-CpA-3'dinucleotides? What is the probability of 5'-TpC-3'dinucleotides?

$$\mathrm{P(CpA)} = \mathrm{P(C)} \times \mathrm{P(A)} = \frac{15}{64} \times \frac{17}{64} = 0.62 \tag{1}$$

$$\mathrm{P(TpC)} = \mathrm{P(C)} \times \mathrm{P(A)} = \frac{17}{64} \times \frac{15}{64} = 0.62 \tag{2}$$

   (b) You find the dinucleotide frequencies don't match the values calculated for independent nucleotides. For instance, the probability of 5'-CpA-3'dinucleotides is 20/256 (7.813%) whereas the probability of 5'-TpC-3'dinucleotides is 15/256 (5.860%). The dinucleotide frequencies overall are given below, in fractions of 256 and as decimal numbers:

			SECOND			
		F	A	C	G	T
1		I	A 17	16	17	18
———	x	R	C 20	15	8	17
256		S	G 15	14	15	16
		T	T 16	15	20	17

		SECOND			
F		A	C	G	T
I	A	0.06641	0.06250	0.06641	0.07031
R	C	0.07812	0.05859	0.03125	0.06641
S	G	0.05859	0.05469	0.05859	0.06250
T	T	0.06250	0.05859	0.07812	0.06641

   i. Verify the single-nucleotide probabilities of A and C in two different ways.

   Examine A and C as both the first and the second nucleotide.

$$\mathrm{P(A)} = \sum_{i \in A,C,G,T} P(A,i) = \frac{17}{64} \tag{3}$$

$$\mathrm{P(A)} = \sum_{i \in A,C,G,T} P(i,A) = \frac{17}{64} \tag{4}$$

$$\mathrm{P(C)} = \sum_{i \in A,C,G,T} P(C,i) = \frac{15}{64} \tag{5}$$

$$\mathrm{P(C)} = \sum_{i \in A,C,G,T} P(i,C) = \frac{15}{64} \tag{6}$$

   ii. What are the probabilities of each nucleotide following a T?

For $i \in A, C, G, T$:

$$\mathrm{P}(i \text{ after T}) = \frac{\mathrm{P(T,\ i)}}{\mathrm{P(T)}} \tag{7}$$

$$\mathrm{P(A \text{ after T})} = \frac{16}{256} \times \frac{64}{17} = 0.24 \tag{8}$$

$$\mathrm{P(C \text{ after T})} = \frac{15}{256} \times \frac{64}{17} = 0.22 \tag{9}$$

$$\mathrm{P(G \text{ after T})} = \frac{20}{256} \times \frac{64}{17} = 0.29 \tag{10}$$

$$\mathrm{P(T \text{ after T})} = \frac{17}{256} \times \frac{64}{17} = 0.25 \tag{11}$$

iii. What are the probabilities of each nucleotide preceding a G?

For $i \in A, C, G, T$:

$$\mathrm{P}(i \text{ before G}) = \frac{\mathrm{P(i,\ G)}}{\mathrm{P(G)}} \tag{12}$$

$$\mathrm{P(A \text{ after G})} = \frac{17}{256} \times \frac{64}{15} = 0.28 \tag{13}$$

$$\mathrm{P(C \text{ after G})} = \frac{8}{256} \times \frac{64}{15} = 0.13 \tag{14}$$

$$\mathrm{P(G \text{ after G})} = \frac{15}{256} \times \frac{64}{15} = 0.25 \tag{15}$$

$$\mathrm{P(T \text{ after G})} = \frac{20}{256} \times \frac{64}{15} = 0.33 \tag{16}$$

iv. What biological phenomenon could explain the dinucleotide patterns you see?

- CpG dinucleotides are less common, while TpG and CpA dinucleotides are more common
- Can arise from cytosine methylation at CpG sites
- Repair of cytosine to uracil deamination is less efficient with 5-methylcytosine, which convert to thymine
- Produces C to T and G to A changes at CpG sites

2. A bag contains two dice. One is a fair die that rolls 1 through 6 with equal probability. The other is a weighted die that has a one-third chance of rolling a 6, and never rolls a 1. You reach in to the bag, pick one of the two dice (either with equal probability), and roll it.

   (a) Write a table of the joint probabilities of picking the fair or the weighted die, and rolling each number.

Joint	1	2	3	4	5	6
Fair	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$
Weighted	0	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{6}$

(b) Compute the conditional probabilities $P(six|fair), P(six|weighted)$.

$$\mathrm{P}(six \,|\, \text{fair}) = \frac{\mathrm{P}(six \,\&\, \text{fair})}{\mathrm{P}(\text{fair})} = \frac{1}{12} \times \frac{2}{1} = \frac{1}{6} \tag{17}$$

$$\mathrm{P}(six \,|\, \text{weighted}) = \frac{\mathrm{P}(six \,\&\, \text{weighted})}{\mathrm{P}(\text{weighted})} = \frac{1}{6} \times \frac{2}{1} = \frac{1}{3} \tag{18}$$

(c) Compute the conditional probabilities $P(fair|six), P(weighted|six)$.

$$\mathrm{P}(\text{fair} \,|\, six) = \frac{\mathrm{P}(six \,\&\, \text{fair})}{\mathrm{P}(six)} = \frac{1}{12} \times \frac{12}{3} = \frac{1}{3} \tag{19}$$

$$\mathrm{P}(\text{weighted} \,|\, six) = \frac{\mathrm{P}(six \,\&\, \text{weighted})}{\mathrm{P}(six)} = \frac{1}{6} \times \frac{12}{3} = \frac{2}{3} \tag{20}$$

Part 3: Maximum Likelihood Estimator

1. The Maximum Likelihood Estimator (MLE) finds the model, or set of parameters, that maximizes the probability of the data. In other words, it maximizes the likelihood of some model θ given the data we obtain and seek to fit a model to. Defining the likelihood as

$$\mathcal{L}(\theta; \mathcal{D}) = p(\text{data} = \mathcal{D} \mid \text{true model} = h_\theta) \tag{1}$$

then the MLE is

$$\hat{\theta}_{MLE} = \arg\max \mathcal{L}(\theta; \mathcal{D}) \tag{2}$$

(a) Given data $\mathcal{D}$ that is a set of outputs $y_1, \ldots, y_n$ arising from inputs $x_1, \ldots, x_n$, write out the MLE above as a probability of the y_is and x_is.

Note: Each output y_i is conditioned on the input x_i (as well as the model)

$$\hat{\theta}_{MLE} = \arg\max p(\text{data} = \mathcal{D} \mid \text{true model} = h_\theta) \tag{3}$$

$$\hat{\theta}_{MLE} = \arg\max p(y_1,\ y_2,\ ...,\ y_n \mid x_1,\ x_2,\ ...,\ x_n,\ h_\theta) \tag{4}$$

(b) The data is related via $y_i = h_\theta(\mathrm{x}_i) + Z_i$ where $h_\theta(\mathrm{x}_i)$ is fixed and Z_i is i.i.d Gaussian, see (3). What is the conditional probability of y_i conditioned on x_i and θ?

$$Z_i \sim \mathcal{N}(0,\ \sigma^2) \tag{5}$$

$$y_i \sim \mathcal{N}(h_\theta(\mathrm{x}_i),\ \sigma^2) \tag{6}$$

(c) Using the answers to (a) and (b), show the MLE estimate $\hat{\theta}_{MLE}$ can be written as follows using the log-likelihood.

$$\hat{\theta}_{MLE} = \arg\min \sum_{i=1}^{n} (y_i - h_\theta(\mathrm{x}_i))^2 \tag{7}$$

$$\hat{\theta}_{MLE} = \arg\max\, p(y_1,\, y_2,\, ...,\, y_n \mid x_1,\, x_2,\, ...,\, x_n,\, h_\theta) \tag{8}$$

$$\hat{\theta}_{MLE} = \arg\max \prod_{i=1}^{n} p(y_i \mid x_i,\, h_\theta) \tag{9}$$

$$\hat{\theta}_{MLE} = \arg\max\ -n \log \sqrt{2\pi}\sigma - \left(\sum_{i=1}^{n} \frac{(y_i - h_\theta(x_i))^2}{2\sigma^2} \right) \tag{10}$$

$$\hat{\theta}_{MLE} = \arg\max\ - \left(\sum_{i=1}^{n} \frac{(y_i - h_\theta(x_i))^2}{2\sigma^2} \right) \tag{11}$$

$$\hat{\theta}_{MLE} = \arg\min \sum_{i=1}^{n} \frac{(y_i - h_\theta(x_i))^2}{2\sigma^2} \tag{12}$$

$$\hat{\theta}_{MLE} = \arg\min \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 \tag{13}$$

Hints:

i. As logs are monotonic functions, taking the log of (2) allows us to find the same optimizer θ

ii. The probabilities of all y_i can be treated as independent, and can be expanded as a product, e.g. $p(y_1,\, y_2) = p(y_1) \cdot (y_2)$

iii. Use the answer from (b) with the formula for a normal distribution

iv. For an optimization problem, constants (added or, if positive, multiplied) don't affect the optimization and can be dropped or ignored

v. The arg max of a term is equivalent to the arg min of the negated term, i.e. $\arg\max x = \arg\min\ -x$

Assignment 1 - Lab Exercises

1. Short Python Practice

Task: Create a class named Dog

- Each instance of the class should have a variable, name (should be set on creating one)
- The class has two main methods, action and age
- Action is an instance method taking in an *optional* parameter, quietly. Calling action prints "WOOF WOOF WOOF " if nothing is passed in, "woof " if quietly is passed in as True. (Use only one hardcoded string in the method)
- Age is a class method that takes in a list of ages in human years and returns it in dog years (Use list comprehension)

In [0]:
```
class Dog:

    def __init__(self, name):
        # complete the class
        self.name = name

    def action(self, quietly=False):
        bark = "woof "
        if quietly:
            print(bark.strip())
        else:
            print((3*bark.upper()).rstrip())

    def age(humans):
        return [i*7 for i in humans]
```

In [0]:
```
human_ages = [3, 4, 7, 4, 10, 6]
```

1. Create an instance of this class, and print its name
2. Call action with both possible options for the parameter
3. Call age on the provided array above

In [0]:
```
# complete the task above
fido = Dog("Fido")
print(fido.name)
fido.action()
fido.action(quietly=True)
Dog.age(human_ages)
```

```
Fido
WOOF WOOF WOOF
woof
```

Out[0]:
```
[21, 28, 49, 28, 70, 42]
```

2. Numpy Practice

Numpy is a commonly used library in Python for handling data, especially helpful for handling arrays/multi-dimensional arrays. It's particularly important for helping bridge the ineffiencies of Python as a high level language with the improved performance of handling data structures in low level languages like C.

Part 1: General Numpy Array Exercises

1. Create a matrix, with shape 10 x 7, of ones
2. Create 10 matrices, with shape 100 x 70, of random integers from -10 to 10
3. Print the number of elements equal between a pair of matrices, for each possible pair in the 10 created above except for with itself (Don't compare the 1st with the 1st)

In [0]:
```
import numpy as np

# example
die_roll = np.random.randint(1, 6)
print("Rolled a " + str(die_roll))
```

```
Rolled a 3
```

In [0]:
```
#complete part 1
uno = np.ones((10,7))
print(uno)
```

```
[[1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1.]]
```

In [0]:
```
mats = []
for i in range(10):
    mats.append(np.random.randint(-10,10,size=(100,70)))
```

In [0]:
```
for i in range(10):
    for j in range(i+1, 10):
        common = np.sum(mats[i]==mats[j])
        print("Matrix "+ str(i)+" and matrix "+str(j)+" have "+ str(common)
        +" elements in common")
```

```
Matrix 0 and matrix 1 have 365 elements in common
Matrix 0 and matrix 2 have 357 elements in common
Matrix 0 and matrix 3 have 374 elements in common
Matrix 0 and matrix 4 have 358 elements in common
Matrix 0 and matrix 5 have 325 elements in common
Matrix 0 and matrix 6 have 311 elements in common
Matrix 0 and matrix 7 have 362 elements in common
Matrix 0 and matrix 8 have 350 elements in common
Matrix 0 and matrix 9 have 326 elements in common
Matrix 1 and matrix 2 have 333 elements in common
Matrix 1 and matrix 3 have 373 elements in common
Matrix 1 and matrix 4 have 335 elements in common
Matrix 1 and matrix 5 have 322 elements in common
Matrix 1 and matrix 6 have 359 elements in common
Matrix 1 and matrix 7 have 342 elements in common
Matrix 1 and matrix 8 have 344 elements in common
Matrix 1 and matrix 9 have 355 elements in common
Matrix 2 and matrix 3 have 346 elements in common
Matrix 2 and matrix 4 have 369 elements in common
Matrix 2 and matrix 5 have 357 elements in common
Matrix 2 and matrix 6 have 369 elements in common
Matrix 2 and matrix 7 have 362 elements in common
Matrix 2 and matrix 8 have 373 elements in common
Matrix 2 and matrix 9 have 342 elements in common
Matrix 3 and matrix 4 have 345 elements in common
Matrix 3 and matrix 5 have 315 elements in common
Matrix 3 and matrix 6 have 378 elements in common
Matrix 3 and matrix 7 have 345 elements in common
Matrix 3 and matrix 8 have 367 elements in common
Matrix 3 and matrix 9 have 331 elements in common
Matrix 4 and matrix 5 have 308 elements in common
Matrix 4 and matrix 6 have 384 elements in common
Matrix 4 and matrix 7 have 357 elements in common
Matrix 4 and matrix 8 have 369 elements in common
Matrix 4 and matrix 9 have 324 elements in common
Matrix 5 and matrix 6 have 339 elements in common
Matrix 5 and matrix 7 have 334 elements in common
Matrix 5 and matrix 8 have 346 elements in common
Matrix 5 and matrix 9 have 366 elements in common
Matrix 6 and matrix 7 have 362 elements in common
Matrix 6 and matrix 8 have 369 elements in common
Matrix 6 and matrix 9 have 348 elements in common
Matrix 7 and matrix 8 have 340 elements in common
Matrix 7 and matrix 9 have 371 elements in common
Matrix 8 and matrix 9 have 308 elements in common
```

Part 2: Splicing and some Numpy Methods

1. Find the pseudoinverse of one of the above matrices (or create a new one with shape 100 x 70)
2. Turns out the last 40 rows and last 10 columns of data were useless. Set a new variable to the matrix used above, without the last 40 rows or 10 columns.
3. Find the inverse for this square matrix.

Optional: You can use the same command for 1 and 3 (briefly explain why if you do)

In [0]:
```
# complete part 2
mat = mats[0]
pinv = np.linalg.pinv(mat)
mat_square = mat[:-40, :-10]
inv = np.linalg.pinv(mat_square)
```

In [0]:
```
print(mat.shape)
print(pinv.shape)
print(mat_square.shape)
print(inv.shape)
```

```
(100, 70)
(70, 100)
(60, 60)
(60, 60)
```

Part 3: Matrix Methods

1. Generate 2 more matrices, *a* and *b* with shape 2 x 3 and 4 x 3.
2. Create 2 matrices, *a_on_b* and *b_on_a* - for the first, stack a on top of b, and the opposite for the second. The join them to create *abba*, with *b_on_a* to the right of *a_on_b*
3. Print the right eigenvalues and eigenvectors for *abba*

In [0]:
```
# complete part 3
a = np.random.randint(-10, 10, size=(2,3))
b = np.random.randint(-10, 10, size=(4,3))
a_on_b = np.vstack((a,b))
b_on_a = np.vstack((b,a))
abba = np.hstack((a_on_b, b_on_a))
print(a)
print(b)
print(abba)
```

```
[[ 2 -5 -8]
 [ 5  2  8]]
[[  6  -3  -1]
 [  6  -7  -4]
 [  7   8   6]
 [  6  -1 -10]]
[[  2  -5  -8   6  -3  -1]
 [  5   2   8   6  -7  -4]
 [  6  -3  -1   7   8   6]
 [  6  -7  -4   6  -1 -10]
 [  7   8   6   2  -5  -8]
 [  6  -1 -10   5   2   8]]
```

In [0]:
```
print(np.linalg.eig(abba))
```

```
(array([ 6.22019279+14.90272269j,  6.22019279-14.90272269j,
        -3.48443388 +9.49878874j, -3.48443388 -9.49878874j,
        -2.01124785 +0.j        ,  8.53973004 +0.j        ]), array([[ 0.4397589
-0.05839195j,  0.4397589 +0.05839195j,
          0.01880205+0.07369621j,  0.01880205-0.07369621j,
         -0.5855294 +0.j        ,  0.21492057+0.j        ],
        [ 0.00403524-0.32072171j,  0.00403524+0.32072171j,
          0.61316062+0.j        ,  0.61316062-0.j        ,
          0.45052556+0.j        , -0.27216933+0.j        ],
        [ 0.02333355-0.47111122j,  0.02333355+0.47111122j,
         -0.34449578+0.22272504j, -0.34449578-0.22272504j,
         -0.19001174+0.j        , -0.21087539+0.j        ],
        [ 0.50277566+0.j        ,  0.50277566-0.j        ,
          0.0073965 -0.00460599j,  0.0073965 +0.00460599j,
          0.59650993+0.j        , -0.4525855 +0.j        ],
        [-0.05409778-0.19424935j, -0.05409778+0.19424935j,
          0.28109597-0.52675325j,  0.28109597+0.52675325j,
          0.21056003+0.j        , -0.5489487 +0.j        ],
        [ 0.24603627-0.35193317j,  0.24603627+0.35193317j,
         -0.30560246-0.00359126j, -0.30560246+0.00359126j,
         -0.13385795+0.j        ,  0.5736604 +0.j        ]]))
```

3. Matplotlib Practice

Matplotlib is a commonly used visualization library.

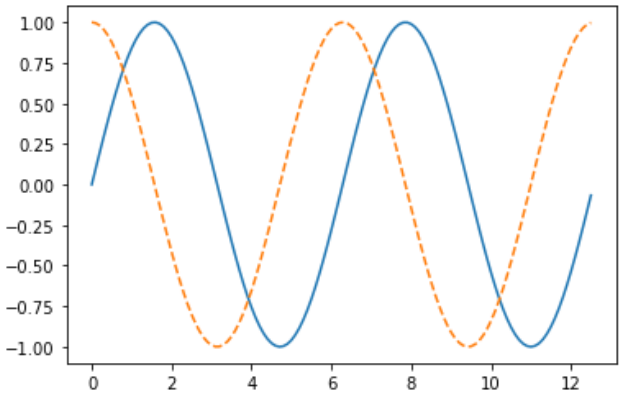
Part 1: Generate plots

1. Generate sine (y_sin) and cosine (y_cos) data for x from 0 to 4 * pi (Use np.arange with step size 0.1)
2. Create a plot with both on the same chart. The sine wave should be solid, cosine dashed
3. Create a second plot with 2 subplots, with the first plotting the sine wave and second plotting the cosine wave

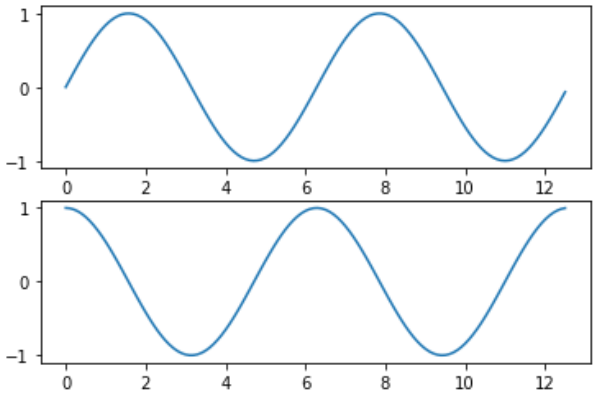
In [0]:
```
import matplotlib.pyplot as plt
```

In [0]:
```
# complete 1
x = np.arange(0, 4*np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
```

In [0]:
```
# complete 2
overlayed = plt.figure()
plt.plot(x, y_sin, "-")
plt.plot(x, y_cos, "--")
plt.show()
```



In [0]:
```
# complete 3
subplotted = plt.figure()
plt.subplot(2,1,1)
plt.plot(x, y_sin)
plt.subplot(2,1,2)
plt.plot(x, y_cos)
plt.show()
```



Part 2: Save Output

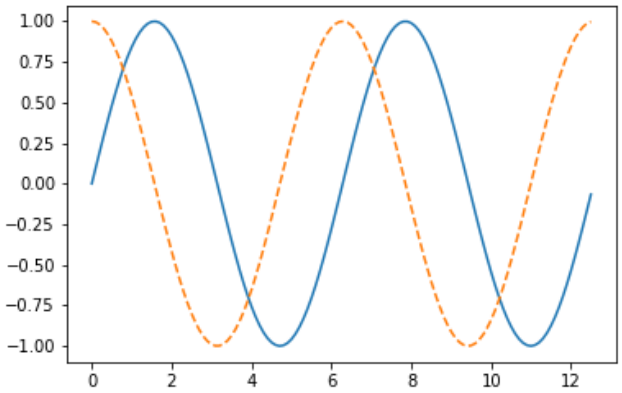
1. For the above plots, save each as a image file.
2. Open both files in this notebook

In [0]:
```
# complete 1
overlayed.savefig("overlayed.png")
subplotted.savefig("subplotted.png")
```

In [0]:
```
from IPython.display import Image
```

In [0]:
```
# complete 2
Image("overlayed.png")
```

Out[0]:



In [0]:
```
Image("subplotted.png")
```

Out[0]: