# Homework Assignment 1

COGS 118A: Introduction to Machine Learning I

**Due: Jan. 15, 2018, 11:59pm (Pacific Time)**

**Instructions:** Please answer the questions below, attach your code, and insert figures to create a PDF file; submit your file to TED (ted.ucsd.edu) by 11:59pm, 1/15/2018. You may search information online but you will need to write code/find solutions to answer the questions yourself.

**Late Policy:** 5% of the total points will be deducted on the first day past due. Every 10% of the total points will be deducted for every extra day past due.

**System Setup:** You can install Anaconda to setup the Jupyter Notebook environment. Most packages are already installed in Anaconda. If some package is not installed, you can use `pip` to install missing package, that is, just type `pip install PACKAGE_NAME` in terminal.

Grade: ____ out of 100 points

## 1 (25 points) Basic Matrix Operations Using NumPy

Suppose $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$, $B = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix}$. We can convert $A$ and $B$ into NumPy arrays by

```
import numpy as np
A = np.array([[1,2], [3,4], [5,6]])
B = np.array([[1,-1], [-1,1], [1,-1]])
```

NumPy package uses `np.dot(A,B)` or `A.dot(B)` to compute the dot product between matrices $A$ and $B$ (Strictly speaking, for 2-D arrays it is equivalent to matrix multiplication, and for 1-D arrays to inner product of vectors). Binary operators such as `*`, `/`, `+` and `-` compute the element-wise operations between two matrices. Please compute the following: (if the matrix multiplication is not possible, write 'impossible' in your answer)

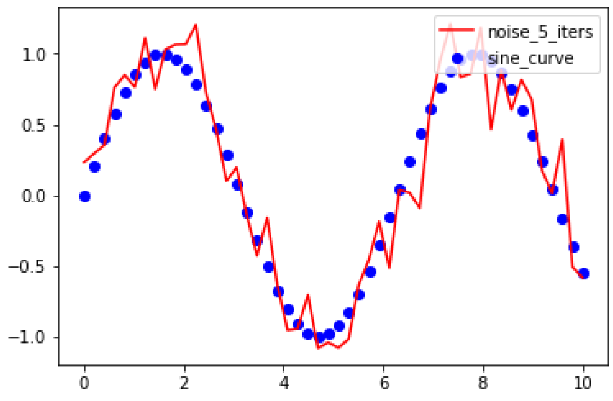
1. $A + B$
2. $A \circ B$ ('$\circ$' operator means element-wise product or Hadamard product)
3. $A^T B$
4. $AB^T$
5. $AB$

## 2 (10 points) Basic Plots Using Matplotlib

Matplotlib is a very useful library to plot graphs. In later course, you may need to plot your own graphs such as: losses v.s. iterations, accuracies v.s. categories and include those graphs in your report. We will start with a simple exercise. Copy and paste the following code into one notebook cell:

```python
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(0)
space = np.linspace(0, 10, num = 50)
sine = np.sin(space)
sine_5 = sine
for i in range(5):
    sine_5 = sine_5 + np.random.normal(scale = 0.1, size = 50)
plt.scatter(space, sine, color = 'b', label = 'sine_curve')
plt.plot(space, sine_5, color = 'r', label = 'noise_5_iters')
plt.legend(loc = 'upper right')
plt.savefig('./Q2.png')
plt.show()
```

The above code creates a graph with the original sine curve and another curve where 5 iterations of Gaussian noise (0 mean, 0.1 standard deviation) are added to the original sine curve. You should get a graph similar to this:



Edit the code above and apply 20 and 100 iterations of Gaussian noise (0 mean, 0.1 standard deviation) on the original sine curve respectively. Plot all four curves in one graph (including the two curves from the original code) and add the legend of the additional curves and label them as `noise_20_iters` and `noise_100_iters`.

## 3 (10 points) Basic Image Operations Using Matplotlib

Matplotlib library also offers a variety of functions for common image processing. In this section you will load the image and show it. First, you need to download the image `cat.jpg` from our course website. The following code can be used to load and display the image.

```python
import matplotlib.pyplot as plt
img = plt.imread("/path/to/image")     # The img here is a NumPy array.
plt.imshow(img)
plt.show()
```

Complete the following questions and paste your output in the homework:

1. Show the `cat.jpg` using the code above.
2. What is the shape of the image array?
   (hint: it is 3-dimensional and you can use `A.shape` to get the shape of array $A$)

## 4 (10 points) Data and Visualization

The dataset description can be found at `https://archive.ics.uci.edu/ml/datasets/Iris` this link. Plot two figures to visualize the dataset using:

1. First 2 feature dimensions.
2. First 3 feature dimensions.

Some useful instructions are shown below:

- Import several useful packages into Python:

```python
import matplotlib.pyplot as plt
from sklearn import datasets
from mpl_toolkits.mplot3d import Axes3D
```

- Load Iris dataset into Python:

```python
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

- Visualize first 2 feature dimensions:

```python
plt.scatter(X[:,0], X[:,1], c = Y, cmap = plt.cm.Paired)
plt.show()
```

- Visualize first 3 feature dimensions:

```
fig = plt.figure(1, figsize = (8, 8))
ax = Axes3D(fig, elev = -150, azim = 110)
ax.scatter(X[:,0], X[:,1], X[:,2], c = Y, cmap = plt.cm.Paired)
plt.show()
```

## 5 (15 points) IBM Watson

Watch a video about IBM Watson: `https://www.youtube.com/watch?v=sgqSatgoTbk` The questions below are supposed to be thought provoking. We will grade your answers based on your general understanding of the problem. At this stage, we are just trying to inspire you to think about the problem. We are not going to be very strict. If your answers are valid based on your assumptions, it is all okay. Your answers don't need to be very long. A few lines should suffice.

1. Imagine you have unlimited computing resources (for example 100,000 computer cores), how would you design your personal "Watson" to perform the similar tasks as shown in the video?

2. What are the possible input data formats of your own "Watson"? Please write down at least three examples.

3. Once you have the input data from the question above, how can you turn them into computable representations for IBM Watson as shown in the video?

4. Name three challenges (potential obstacles) to build your own "Watson"?

## 6 (30 points) Data Manipulation

We have already had a glimpse of Iris dataset in Question 4. In this question, we still use the same Iris dataset. In fact, you can find the shape of array $X$ is (150, 4) by running `X.shape`, which means there are 150 data points and each data point has 4 features. Here we will calculate some properties of the array $X$ and perform some basic data manipulation:

1. Show first 3 features of first 5 data points (first 3 columns and first 5 rows) of array $X$.

2. Calculate mean and variance of the 3rd feature (the 3rd column) of array $X$.

3. Given a weight vector $\mathbf{w} = (1, 2, 3, 4)$, perform a linear projection on the 4 features of all data points using the weight vector $\mathbf{w}$, that is, do a dot product between the 4 features of all data points and the weight vector $\mathbf{w}$. The shape of projected data points should be (150, 1) or (150,), which depends on the weight vector is regarded as matrix or vector. Please calculate the mean of the projected data points.
   (hint: one line code using `np.dot` is enough to calculate the projection, but you should pay attention to the dimension match in dot product)

4. Randomly sample 4 data points (rows) of array $X$ by randomly choosing index. Show the index and the sampled data points.

5. Add one more feature (one more column) to the array $X$ after the last feature. The value of the added feature is constant 1 for all data points. Then show the first data point (first row) of the new array.

Some useful instructions are shown below:

- Get a row or a column of the array $X$:

```
print X[0]           # Print the first row of array X.
print X[:, 0]        # Print the first column of array X.
                     # ':' here means all rows and '0' means column 0.
print X[:, 0].mean() # Print the mean of first column.
print X[:, 0].var()  # Print the variance of first column.
```

- Generate a column of array $X$:

```
X0 = np.zeros((150, 1)) # Generate an array of 0 with shape (150, 1).
X1 = np.ones((150, 1))  # Generate an array of 1 with shape (150, 1).
```

- Add one more column:

```
newX = np.hstack((X, X1)) # Stack the arrays horizontally,
                          # which behaves like add column(s).
```

- Get part of the array:

```
print X[3:5, 1:3] # Print 3rd and 4th rows, 2nd and 3rd columns.
print X[:3, :2]   # Print first 3 rows, first 2 columns.
```

- Randomly sample several data points (rows).

```
index = np.random.randint(0,150,3) # Randomly choose 3 integers in
                                   # interval [0, 150) and save as index.
print index                        # Print index.
print X[index]                     # Print the 3 rows based on index.
```