Assignment 1

1. Vector Calculus Review (15 pts)

Let $x, c \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$. For the following parts, before taking any derivatives, identify what the derivative looks like (is it a scalar, vector, or matrix?) and how we calculate each term in the derivative. Then carefully solve for an arbitrary entry of the derivative, then stack/arrange all of them to get the final result. Note that the convention we will use going forward is that vector derivatives of a scalar (with respect to a column vector) are expressed as a row vector, i.e. $\frac{\partial f}{\partial x} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right]$ since a row acting on a column gives a scalar. You may have seen alternative conventions before, but the important thing is that you need to understand the types of objects and how they map to the shapes of the multidimensional arrays we use to represent those types.

(a) Show $\frac{\partial}{\partial x}(x^T c) = c^T$

(b) Show $\frac{\partial}{\partial x}\|x\|_2^2 = 2x^T$

(c) Show $\frac{\partial}{\partial x}(Ax) = A$

(d) Show $\frac{\partial}{\partial x}(x^T A x) = x^T(A + A^T)$

(e) Under what condition is the previous derivative equal to $2x^T A$?

2. Bayes' Rule (10 pts)

Assume the probability of a certain disease is 0.01. The probability of test positive given that a person is infected with the disease is 0.95 and the probability of test positive given the person is not infected with the disease is 0.05.

(a) Calculate the probability of test positive.

(b) Use Bayes' Rule to calculate the probability of being infected with the disease given that the test is positive.

3. Gradient Descent Mechanics (20 pts)

Gradient descent is the primary algorithm to search optimal parameters for our models. Typically, we want to solve optimization problems stated as

$$\min_{\theta\in\Theta} \mathcal{L}(f_\theta, \mathcal{D})$$

where $\mathcal{L}$ are differentiable functions. In this example, we look at a simple supervised learning problem where given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, we want to find the optimal parameters θ that minimizes some loss. We consider different models for learning the mapping from input to output, and examine the behavior of gradient descent for each model.

(a) The simplest parametric model entails learning a single-parameter constant function, where we set $\hat{y}_i = \theta$. We wish to find

$$\hat{\theta}_{const} = \min_{\theta\in\mathbb{R}} \mathcal{L}(f_\theta, \mathcal{D}) = \min_{\theta\in\mathbb{R}} \frac{1}{N}\sum_{i=1}^{N}(y_i - \theta)^2$$

i. What is the gradient of $\mathcal{L}$ with respect to θ?

ii. What is the optimal value of θ?

iii. Write the gradient descent update.

iv. Stochastic Gradient Descent (SGD) is an alternative optimization algorithm, where instead of using all N samples, we use single sample per optimization step to update the model. What is the contribution of each data-point to the full gradient update?

(b) Instead of constant functions, we now consider a single-parameter linear model $\hat{y}_i(x_i) = \theta(x_i)$ where we search for θ such that

$$\hat{\theta} = \min_{\theta \in \mathbb{R}} \frac{1}{N} \sum_{i=1}^{N} (y_i - \theta x_i)^2$$

i. What is the gradient of $\mathcal{L}$ with respect to θ?
ii. What is the optimal value of θ?
iii. Write the gradient descent update.
iv. Do all points get the same vote in the update? Why or why not?

4. MAP Interpretation of Ridge Regression (20 pts)

Consider the Ridge Regression estimator (Hint: here the X is feature matrix, equivalent to X^T in our slides)

$$\arg\min_{w} \|X\boldsymbol{w} - y\|^2 + \lambda \|\boldsymbol{w}\|^2$$

We know this is solved by

$$\hat{\boldsymbol{w}} = (X^T X + \lambda I)^{-1} X^T y$$

One interpretation of Ridge Regression is to find the Maximum A Posteriori (MAP) estimate on $\boldsymbol{w}$, the parameters, assuming that the prior of w is $N(0, I)$ and that the random Y is generated using

$$Y = X\boldsymbol{w} + \sqrt{\lambda} N$$

Note that each entry of vector N is zero-mean, unit-variance normal. Show that $\hat{\boldsymbol{w}} = (X^T X + \lambda I)^{-1} X^T y$ is indeed the MAP estimate for $\boldsymbol{w}$ given an observation on $Y = y$.

5. Programming (35 pts)

In this lab, please write your code in Jupyter Notebook, take screenshots of the **code blocks and their outputs together**, and then paste the screenshots after your answers of Q1-Q4. You also need to submit your .ipynb file.

Lab 1 Essential Libraries and Tools

- ## NumPy

EN tutorial: https://www.w3schools.com/python/numpy/numpy_intro.asp

CN tutorial: https://www.runoob.com/numpy/numpy-tutorial.html

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for **Numerical Python**.

Task 1: In the code cell below, complete the function `insertSecond` which takes two arguments, array `a` and element `b`, and returns an array which `b` is inserted before the second element in `a`.

```python
# You should return your result.
import numpy as np
def insertSecond(a, b):
    # YOUR CODE HERE
    raise NotImplementedError()

# Test cases
assert np.array_equal(insertSecond(np.array([-5,-10,-12,-6]),5), np.array([-5, 5, -10,
-12, -6]))
assert np.array_equal(insertSecond(np.array([1,2,3]),7), np.array([1, 7, 2, 3]))
assert np.array_equal(insertSecond(np.array([-5,-10,-12,-6]),8), np.array([ -5, 8, -10,
-12, -6]))
assert np.array_equal(insertSecond(np.array([1,2,3]),12), np.array([1, 12, 2, 3]))
```

Task 2: In the code cell below, complete the function `mergeArrays` takes two array arguments `a` and `b`. The two arrays are merged, in which process the duplicate elements are removed. Finally the merged array are sorted.

```
# You should return your result.
import numpy as np
def mergeArrays(a,b):
    # YOUR CODE HERE
    raise NotImplementedError()

# Test cases
assert np.array_equal(mergeArrays(np.array([1,1,4,8,1]), np.array([2, 3])),
np.array([1, 2, 3, 4, 8]))
assert np.array_equal(mergeArrays(np.array([-5,-10,-10,-6]), np.array([-5, 8, -10, -12,
-6])),np.array([-12, -10, -6, -5, 8]) )
assert np.array_equal(mergeArrays(np.array([1,1,6,8,1]), np.array([2, 3])),
np.array([1, 2, 3, 6, 8]))
```

- ## SciPy

EN tutorial: https://www.w3schools.com/python/scipy/scipy_intro.php

CN tutorial: https://www.runoob.com/scipy/scipy-tutorial.html

SciPy is a scientific computation library that uses NumPy underneath.

SciPy stands for **Scientific Python**.

It provides more utility functions for optimization, stats and signal processing.

Like NumPy, SciPy is open source so we can use it freely.

SciPy was created by NumPy's creator Travis Olliphant.

- ## matplotlib

EN tutorial: https://www.w3schools.com/python/matplotlib_intro.asp

CN tutorial: https://www.runoob.com/matplotlib/matplotlib-tutorial.html

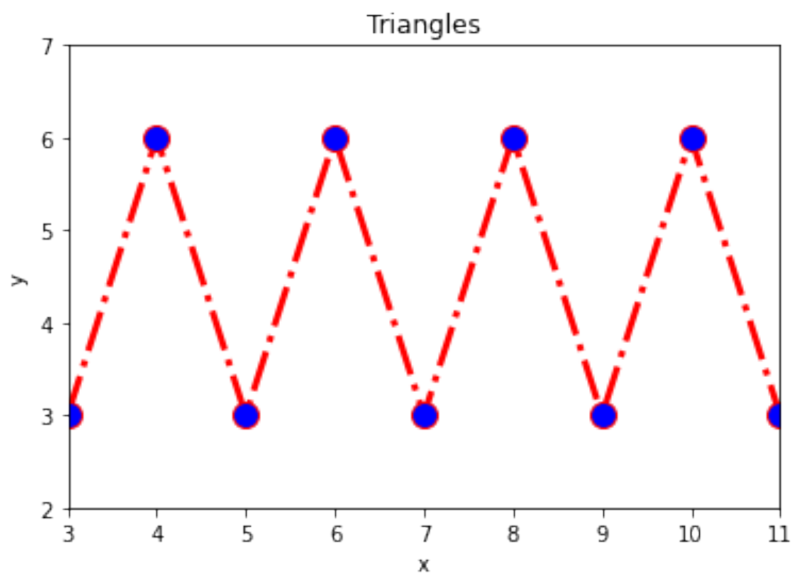
Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib was created by John D. Hunter.

Matplotlib is open source and we can use it freely.

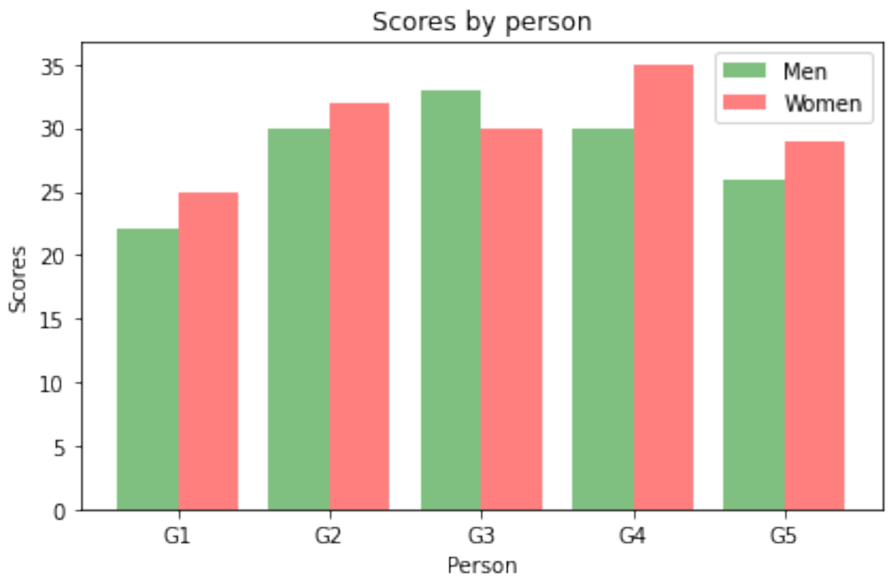
Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Complete the code below to create a image which looks exactly the same as the follow one:



```python
import matplotlib.pyplot as plt

# x axis values
x = [3,4,5,6,7,8,9,10,11]
# y axis values
y = [3,6,3,6,3,6,3,6,3]

# plotting the points
plt.plot(x, y, color='red', linestyle='dashdot', linewidth = 3,
         marker='o', markerfacecolor='blue', markersize=12)
#Set the y-limits of the current axes.
plt.ylim(2,7)
#Set the x-limits of the current axes.
plt.xlim(3,11)

# naming the x axis
plt.xlabel('x')
# naming the y axis
plt.ylabel('y')

# giving a title to my graph
plt.title('Triangles')

plt.show()
```

Task 3: Complete the code below to create a image which looks exactly the same as the follow one:



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

# data to plot
n_groups = 5
men_means = (22, 30, 33, 30, 26)
women_means = (25, 32, 30, 35, 29)
alpha = 0.5
# YOUR CODE HERE
raise NotImplementedError()
plt.show()
```

- ## Pandas

EN tutorial: https://www.w3schools.com/python/pandas/pandas_intro.asp

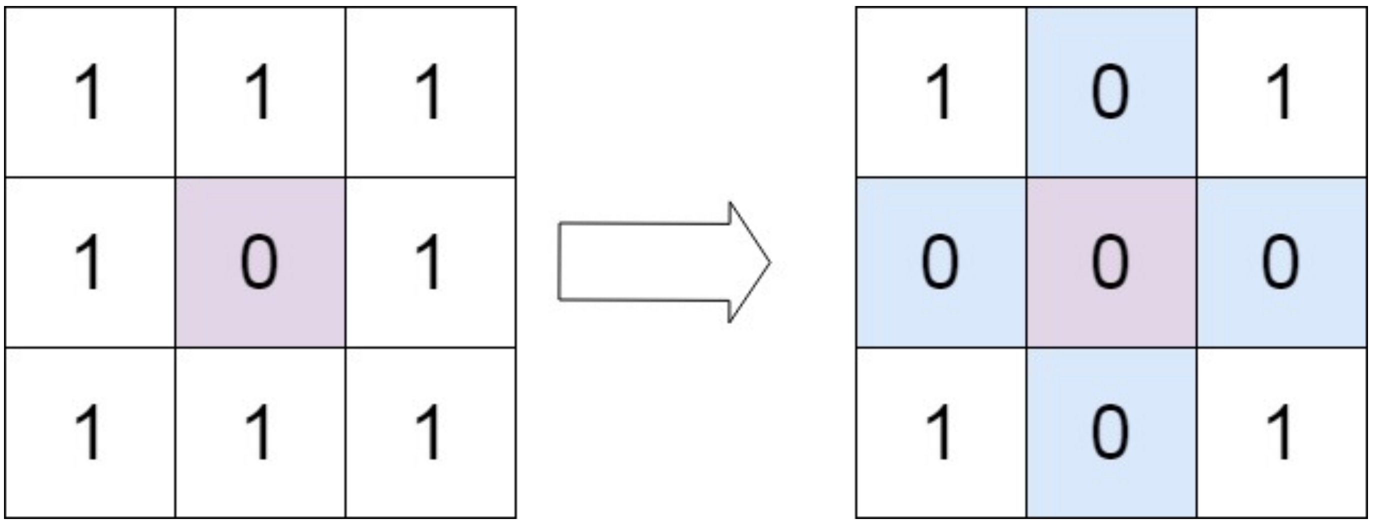
CN tutorial: https://www.runoob.com/pandas/pandas-tutorial.html

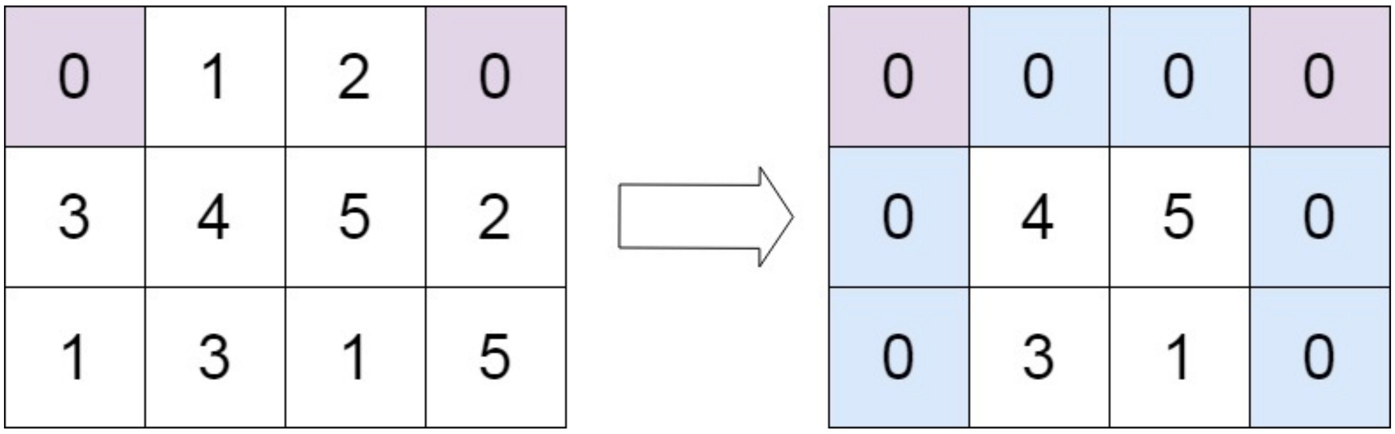
Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Task 4: In the code cell below, complete the function `setDataFrameZeros` which takes a dataFrame `df` as an arguement, and returns a new dataFrame. The label and index in the new dataFrame is the same as those in `df`. If an element is 0 in `df`, set its entire row and column to 0 in the new dataFrame. All other elements are the same as those in `df`.





```python
# You should return your result.
import pandas as pd
def setDataFrameZeros(df):
    # YOUR CODE HERE
    raise NotImplementedError()

# Test cases
df1 = pd.DataFrame({'c1': [1, 4, 7],
                    'c2': [2, 0, 8],
                    'c3': [3, 6, 9]})
df2 = pd.DataFrame({'c1': [1, 0, 7],
                    'c2': [0, 0, 0],
                    'c3': [3, 0, 9]})
assert (df2.equals(setDataFrameZeros(df1)))

df1 = pd.DataFrame({'c1': [0, 3, 1],
                    'c2': [1, 4, 3],
                    'c3': [2, 5, 1],
                    'c4': [0, 2, 5]})
df2 = pd.DataFrame({'c1': [0, 0, 0],
                    'c2': [0, 4, 3],
                    'c3': [0, 5, 1],
                    'c4': [0, 0, 0]})
assert (df2.equals(setDataFrameZeros(df1)))
```

```python
df1 = pd.DataFrame({'c1': [1, 4, 7],
                    'c2': [2, 0, 8],
                    'c3': [3, 6, 9]})
df2 = pd.DataFrame({'c1': [1, 0, 7],
                    'c2': [0, 0, 0],
                    'c3': [3, 0, 9]})
assert (df2.equals(setDataFrameZeros(df1)))

df1 = pd.DataFrame({'c1': [0, 3, 1],
                   'c2': [1, 4, 3],
                   'c3': [2, 5, 1],
                   'c4': [0, 2, 5]})
df2 = pd.DataFrame({'c1': [0, 0, 0],
                   'c2': [0, 4, 3],
                   'c3': [0, 5, 1],
                   'c4': [0, 0, 0]})
assert (df2.equals(setDataFrameZeros(df1)))
```