HW#0-STA208-S24: Getting Started / Warmup Exercises

Part I: (Do not turn in)

Basics of how to work with jupyter notebooks and colab.

Colab/notebooks:

- colab and google drive
- code cells / text cells
- save (and pin) notebooks
- markdown
- latex

- I/O (see below; reference: https://colab.research.google.com/notebooks/io.ipynb)

Resources:

Very basics of LATEX

Overview of Colaboratory

Guide to Markdown

Saving and loading notebooks in GitHub

I/O on Colab and using bash commands (this section is also in the notebook "lecture1")

To upload a file to Colab from a local drive, run the following code (you will be prompted to select the file);

```
In [ ]: from google.colab import files
        uploaded = files.upload()

        # the below shows that I have imported the file "winequality-red.csv" (to do this c
        # on the prompt "Choose Files" and select from your loval machine)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 1
----> 1 from google.colab import files
      2 uploaded = files.upload()

ModuleNotFoundError: No module named 'google'
```

In [ ]:

```
```

- Alternatively, you can also mount your entire google drive, and then have access to all the files there; after mounting google drive there is no need to individually upload files;
- To mount google drive, you can either click on the corresponding "mount" icon in the left column under 'Files', or run the next code cell, or use below code;
- then run the subsequent code cell to load the `winequality' data

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:

```
## Lines that start with ! run a bash (shell) command

# If mounting google drive, run the following to see whether the winedata file is a
# !ls -l /content/drive/MyDrive/data/winequality-red.csv

# if uploading the data from a local drive, run the following to see whether the wi
!ls -l winequality-red.csv
```

```
-rw-r--r-- 1 root root 84199 Apr  1 16:26 winequality-red.csv
```

In [ ]:

```
# If google drive has been mounted, run the following to see the header of the file
# !head /content/drive/MyDrive/data/winequality-red.csv

# If data file has been loaded from a local drive, run the following to see the hea
!head winequality-red.csv
```

```
"fixed acidity";"volatile acidity";"citric acid";"residual sugar";"chlorides";"free
sulfur dioxide";"total sulfur dioxide";"density";"pH";"sulphates";"alcohol";"qualit
y"
7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5
7.8;0.88;0;2.6;0.098;25;67;0.9968;3.2;0.68;9.8;5
7.8;0.76;0.04;2.3;0.092;15;54;0.997;3.26;0.65;9.8;5
11.2;0.28;0.56;1.9;0.075;17;60;0.998;3.16;0.58;9.8;6
7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5
7.4;0.66;0;1.8;0.075;13;40;0.9978;3.51;0.56;9.4;5
7.9;0.6;0.06;1.6;0.069;15;59;0.9964;3.3;0.46;9.4;5
7.3;0.65;0;1.2;0.065;15;21;0.9946;3.39;0.47;10;7
7.8;0.58;0.02;2;0.073;9;18;0.9968;3.36;0.57;9.5;7
```

Review basics of python, numpy, pandas

(official Python documentation: https://docs.python.org/3/)

- lists, tuples, arrays (perhaps dictionaries, sets)
- functions
- if else; for loops
- [f(a) for a in L] list comprehension (e.g. see https://www.pythoncheatsheet.org/cheatsheet/comprehensions)
- copying vs. referencing
- mutable - immutable (e.g. see https://realpython.com/python-mutable-vs-immutable-types/)

- slicing a numpy array
- pandas dataframes (e.g. see https://pandas.pydata.org/docs/reference/frame.html)
- slicing a pandas dataframe
- subsetting using iloc and loc
- subsetting using criteria
- fit(), predict(), transform() (fit_predict(), fit_transfrom()) methods

General resources:

- https://www.pythoncheatsheet.org/
- https://pandas.pydata.org/docs/reference/io.html

Part II: Some Exercises (turn in for credit: due on April 10; upload as pdf to Gradescope; due Wed. April 10, 11:59pm)

These exercises are meant to recall basic geometric interpretation underlying concepts from linear algebra and to get some practice in using basic numpy/python.

1) Add a new text cell to this notebook right below the fomulation of this problem 1 (in particular, split the cell - see keyboard shortcuts in the tools dropdown menu for the command) and add your answers to this problem there. Use LATEX for writing formulas.

Suppose u is a d-dimensional vector that is normal to a hyperplane H. Consider the $(d \times d)$ -matrix

$$U := I_d - \frac{1}{\|u\|^2} u u^\top .$$

a) Show that for any d-dimensional vector v, the vector Uv (matrix multiplication) lies in H. [HINT: Show this by using the property of a normal vector.]

b) Provide a geometric interpretation of the vector Uv (i.e. describe how it relates geometrically to v).

c) With I_d denoting the $(d \times d)$ identity matrix, the matrix

$$U^* := I_d - \frac{2}{\|u\|^2} u u^\top$$

is called *elementary reflector* (or sometimes *Hausholder transformation*). Compare the vector U^*v to Uv and explain the name *elementary reflector*. [HINT: Drawing a figure for $d = 2$ might help.]

1. Answers

a)

If u is normal to a hyperplane, and Uv is a vector on the plane then <Uv,u> = 0

$$(I_d - \frac{1}{\|u\|^2} uu^\top)v$$

distribute v

$$(v - \frac{1}{\|u\|^2} uu^\top v)$$

Note $uu^\top$ is a projection matrix that projects onto u, $uu^\top v$ is a scalar multiple of u

$$(v - \frac{u^\top v}{\|u\|^2} u)$$

This is a projection of v onto u because of that the equation above will lie on H because the equation is orthogonal to u

b)

U is a projection matrix that is orthogonal to the vector u, therefore any vector v that is linearly transformed using the matrix U (Uv) is also orthogonal to u. Given that u is orthogonal to the hyperplane H, Uv would therefore be on the hyperplane H as Uv is orthgonal to u.

c)

Uv is the projection of v onto the hyperplane of H. U*v is the "reflection" of v using Uv as the base of reflection.

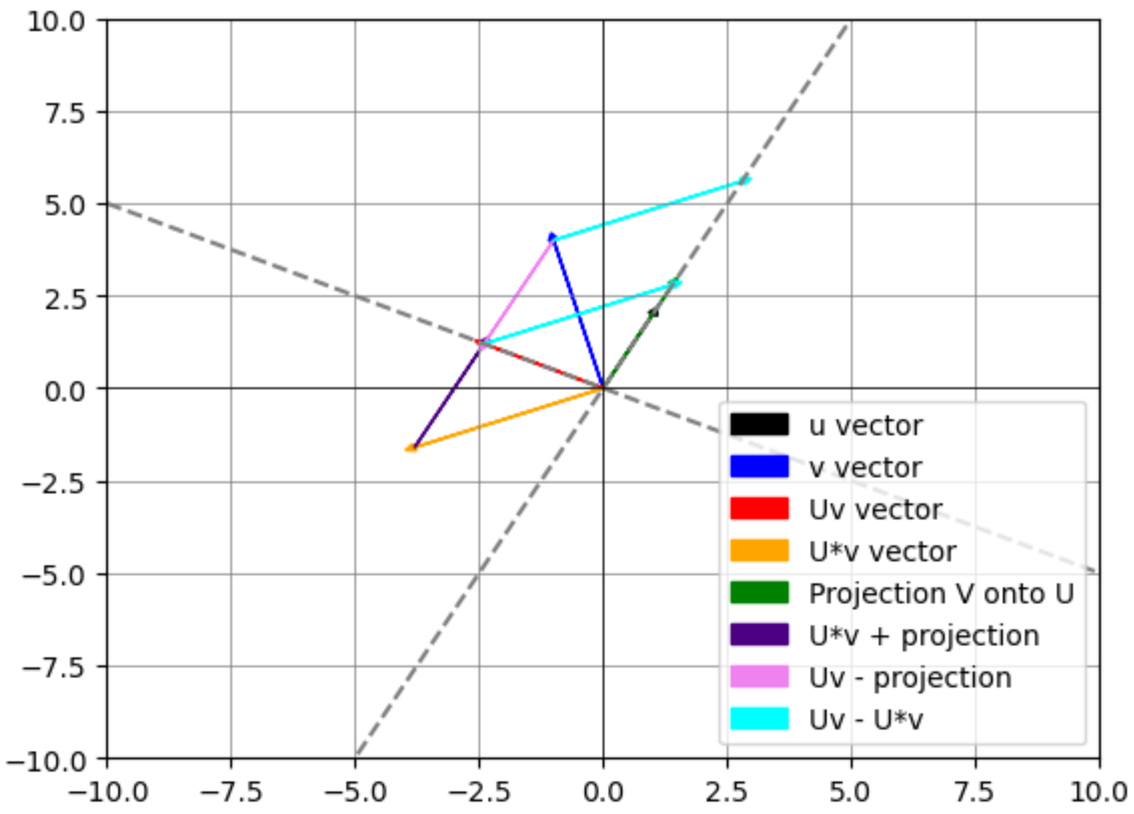
```
In [ ]:
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
def arrow(origin, vector, c, l):
    return plt.arrow(origin[0], origin[1], vector[0], vector[1], color=c, label=l,


#Create all needed vectors
u = np.array([1,2])
v = np.array([-1,4])
U = np.eye(2) - np.outer(u,u)/(np.linalg.norm(u)**2)
U_star = np.eye(2) - 2* np.outer(u,u)/(np.linalg.norm(u)**2)
Uv = U.dot(v)
U_star_v = U_star.dot(v)
proj = np.outer(u,u).dot(v)/(np.linalg.norm(u)**2)
min_proj = -1 * proj
#set up H hyperplane
x = np.linspace(-10,10,100)
y = -0.5*x

u_span = [x*u[0], x*u[1]]
#Create graph
origin = np.array([0,0])

u_vector = arrow(origin, u, 'black', 'u vector')
v_vector = arrow(origin, v, 'blue', 'v vector')
Uv_vector = arrow(origin, Uv, 'red', 'Uv vector')
U_star_v_vector = arrow(origin, U_star_v, 'orange', 'U*v vector')
proj_vector = arrow(origin, proj, 'green', 'Projection V onto U')
reflect = arrow(U_star_v, proj, "indigo", "U*v + projection")
non_reflect = arrow(v, min_proj, 'violet', 'Uv - projection')
Uv_USv = arrow(Uv, -1 * U_star_v, 'cyan', 'Uv - U*v')
v_USv = arrow(v, -1*U_star_v, 'cyan','')
plt.plot(x, y, color='gray', linestyle = '--', label="H hyperplane")
plt.plot(u_span[0], u_span[1], color='gray', linestyle='--', label='u span')
plt.axhline(0,color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(color='gray', linewidth=0.5)
plt.legend(handles=[u_vector, v_vector, Uv_vector, U_star_v_vector, proj_vector, re
plt.xlim(-10,10)
plt.ylim(-10,10)
plt.show()
```



2) Use numpy to do the computations in this problem. To this end, use this notebook, add a code cell right below the formulation of this problem and put your solutions there.

a) Define $u = (1, 2, 3, 3, 2, 1)$, $v = (1, 2, 3, 4, 5, 6)$.

- Compute the matrix $U = I - \frac{1}{\|u\|^2} uu^\top$ and find Uv (print both, the matrix U and the vector Uv)
- Compute $U^* = I - \frac{2}{\|u\|^2} uu^\top$ and find $U^* v$ (print both, the matrix U^* and the vector $U^* v$).

b) Consider the real value $\alpha = \frac{\langle u, Uv - U^* v\rangle}{\|u\| \cdot \|Uv - U^* v\|}$.

- Make a conjecture of what the value of α is (use your geometric intuition - again, drawing a figure in $d = 2$ should help).
- Compute α, and explain the value you obtain.

In [ ]:
```
u = np.array([1,2,3,3,2,1])
v = np.array([1,2,3,4,5,6])

U = np.eye(6) - np.outer(u,u)/np.linalg.norm(u)**2
Uv = U.dot(v)
print(f"U = {U} and Uv = {Uv}")
U_star = np.eye(6) - 2*(np.outer(u,u)/np.linalg.norm(u)**2)
USv = U_star.dot(v)
print(f"U* = {U_star} and U*v = {USv}")
```

```
U = [[ 0.96428571 -0.07142857 -0.10714286 -0.10714286 -0.07142857 -0.03571429]
 [-0.07142857  0.85714286 -0.21428571 -0.21428571 -0.14285714 -0.07142857]
 [-0.10714286 -0.21428571  0.67857143 -0.32142857 -0.21428571 -0.10714286]
 [-0.10714286 -0.21428571 -0.32142857  0.67857143 -0.21428571 -0.10714286]
 [-0.07142857 -0.14285714 -0.21428571 -0.21428571  0.85714286 -0.07142857]
 [-0.03571429 -0.07142857 -0.10714286 -0.10714286 -0.07142857  0.96428571]] and Uv
= [-0.5 -1.  -1.5 -0.5  2.   4.5]
U* = [[ 0.92857143 -0.14285714 -0.21428571 -0.21428571 -0.14285714 -0.07142857]
 [-0.14285714  0.71428571 -0.42857143 -0.42857143 -0.28571429 -0.14285714]
 [-0.21428571 -0.42857143  0.35714286 -0.64285714 -0.42857143 -0.21428571]
 [-0.21428571 -0.42857143 -0.64285714  0.35714286 -0.42857143 -0.21428571]
 [-0.14285714 -0.28571429 -0.42857143 -0.42857143  0.71428571 -0.14285714]
 [-0.07142857 -0.14285714 -0.21428571 -0.21428571 -0.14285714  0.92857143]] and U*v
= [-2. -4. -6. -5. -1.  3.]
```

considering the real value $\alpha = \frac{<u, Uv - U*v>}{||u|| * ||Uv - U*v||}$

since a = Uv-U*v lies on u, the formula alpha most likely is 1

```
In [ ]: s = Uv-USv
        alpha = u.dot(s)/(np.linalg.norm(u)*np.linalg.norm(s))
        print(f"alpha = {alpha}")
```

```
alpha = 1.0
```

3) Consider the matrix $UU^\top$ with U from above. We are now concerned about the eigenvalues of $UU^\top$.

- Make a conjecture about the values of the eigenvalues of $UU^\top$ (positive, negative, zero, certain specific values??).
- Compute the eigenvalues using numpy and iterprete the values that you find.
- Explain why $UU^\top = UU = U$, i.e. U is idempotent.
- Let V be the (6×3)-matrix consisting of the first three columns of U. Is V also idempotent? Explain.

The eigenvalues of UU^T is most likely either 0 or positive because UU^T is positive semi-definite

```
In [ ]: UUt = np.matmul(U, (np.transpose(U)))
        ev = np.linalg.eigvals(UUt)
        ev * np.eye(6)
```

```
Out[ ]: array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                0.00000000e+00, 0.00000000e+00],
               [0.00000000e+00, 3.05321739e-17, 0.00000000e+00, 0.00000000e+00,
                0.00000000e+00, 0.00000000e+00],
               [0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 0.00000000e+00,
                0.00000000e+00, 0.00000000e+00],
               [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.00000000e+00,
                0.00000000e+00, 0.00000000e+00],
               [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                1.00000000e+00, 0.00000000e+00],
               [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
                0.00000000e+00, 1.00000000e+00]])
```

$$U = I - \frac{1}{\|u\|^2}uu^\top$$

$$UU^\top = (I - \frac{1}{\|u\|^2}uu^\top)(I - \frac{1}{\|u\|^2}uu^\top)^\top$$

$$UU^\top = (I - \frac{1}{\|u\|^2}uu^\top)(I - \frac{1}{\|u\|^2}uu^\top)$$

$$UU^\top = I - \frac{2}{\|u\|^2}uu^\top + \frac{1}{\|u\|^2}uu^\top$$

$$UU^\top = I - \frac{1}{\|u\|^2}uu^\top$$

Recall $U = I - \frac{1}{\|u\|^2}uu^\top$ and since $U^\top = U$,

$$UU^\top = UU = U$$

By definition, an idempotent matrix must be a square matrix, therefore since V is (6X3) which is not a square matrix, it is not idempotent.

4) Some numpy practice: With $U = (U_{ij})_{i,j=1,...,6}$ from above do the following:

- Find the largest and smallest off-diagonal elements of $UU^\top$.
- Find the largest and smallest diagonal elements of $UU^\top$.
- Compute the value $max_i \sum_j |U_{ij}|$.
- Print the elements of the second column of $UU^\top$ below the diagonal.
- Verify idempotence of U by showing that $UU - U = 0$ (where the 0 on the right hand side is the 0-matrix).

In [ ]:
```
#Since UU^T is symmetric, we only need to look at the upper triangle for the off-di
off_diag = UUt[np.triu_indices(len(UUt), k=1)]
max_off = np.max(off_diag)
min_off = np.min(off_diag)

print(f"The maximum off-diagonal element of UUt is {max_off}")
print(f"The minimum off-diagonal element of UUt is {min_off}")

#Get the diagonal elements
diag = np.diag(UUt)
max_diag = np.max(diag)
min_diag = np.min(diag)

print(f"The maximum diagonal element of UUt is {max_diag}")
print(f"The minimum diagonal element of UUt is {min_diag}")

#Compute max of the sum of the rows
max_row = np.max(np.sum(abs(UUt), axis=0))
print(f"The max row sum in the matrix is {max_row}")

#Get the elements of the second column of UUt below the diagonal
print(f"The elements of the second column of UUt below the diagonal are {UUt[2:,1]}


print(f"The matrix UU - U = {np.matmul(U,U) - U}")
```

```
The maximum off-diagonal element of UUt is -0.03571428571428572
The minimum off-diagonal element of UUt is -0.32142857142857145
The maximum diagonal element of UUt is 0.9642857142857143
The minimum diagonal element of UUt is 0.6785714285714285
The max row sum in the matrix is 1.6428571428571428
The elements of the second column of UUt below the diagonal are [-0.21428571 -0.214
28571 -0.14285714 -0.07142857]
The matrix UU - U = [[ 1.11022302e-16 -2.77555756e-17 -2.77555756e-17 -1.38777878
e-17
  -2.77555756e-17 -1.38777878e-17]
 [-2.77555756e-17  0.00000000e+00 -5.55111512e-17 -5.55111512e-17
  -5.55111512e-17 -2.77555756e-17]
 [-2.77555756e-17 -5.55111512e-17 -1.11022302e-16  0.00000000e+00
  -5.55111512e-17  0.00000000e+00]
 [-1.38777878e-17 -5.55111512e-17  0.00000000e+00  0.00000000e+00
  -5.55111512e-17 -2.77555756e-17]
 [-2.77555756e-17 -5.55111512e-17  0.00000000e+00  0.00000000e+00
   0.00000000e+00 -1.38777878e-17]
 [-6.93889390e-18 -1.38777878e-17  1.38777878e-17  0.00000000e+00
  -1.38777878e-17  0.00000000e+00]]
```