CMSC 25300/35300, STAT 27700 (Spring 2025)
Homework 3

Submission Instructions:

- Please submit your homework in PDF to Gradescope (which can be accessed from the course's website on Canvas);
- Please attach your code (either as code blocks or as a .ipynb jupyter notebook converted to a pdf) to the end of your answer document. A template will be provided to do this via Latex.
- Note that you do not need to copy the problem statements in your solution, *as long as you clearly indicate the problem numbers* (e.g., 1.a, 2.c, etc).

Note on coding problems: library functions that essentially do the problem for you (e.g., `np.linalg`'s `lstsq` or `qr`) should not be used for this class. Functions like `np.linalg`'s `inv` and `solve` are acceptable since we are not as focused on the mechanics of solving systems of equations – you can ask if you are unsure if a function is allowed for a given question.

1. **Orthogonal columns**. Consider the matrix and vector

$$\boldsymbol{X} = \begin{bmatrix} 6 & 7 \\ 8 & 1 \\ 0 & 5 \end{bmatrix} \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} 1 \\ 6 \\ -2 \end{bmatrix} .$$

   a) (7 points) By hand, find two orthonormal vectors that span the plane spanned by columns of $\boldsymbol{X}$.

   b) (6 points) We know that the least square (LS) estimate is $\hat{\boldsymbol{y}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$, and this is closely related to projection matrices. In the previous problem, we computed $\boldsymbol{U}$, the orthogonalization of $\boldsymbol{X}$. Explain intuitively why we can also use the orthogonalization to compute the LS estimate: $\hat{\boldsymbol{y}} = \boldsymbol{U}(\boldsymbol{U}^T\boldsymbol{U})^{-1}\boldsymbol{U}^T\boldsymbol{y}$.

   c) (7 points) Find $\hat{\boldsymbol{y}}$ using the expression in terms of $\boldsymbol{U}$. Explain why using the orthogonalization of the columns of $\boldsymbol{X}$ makes computing the LS estimate easier than using the original $\boldsymbol{X}$.

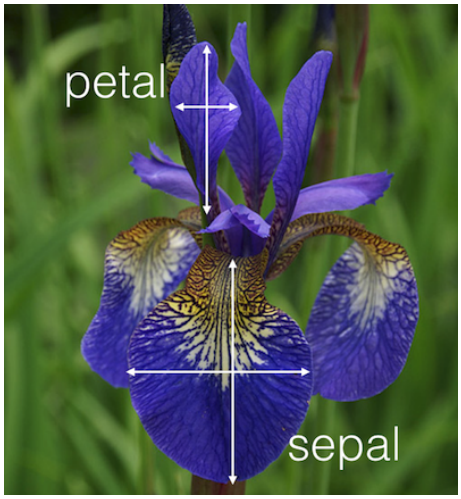
2. **Projection matrix**. Define a subspace $\mathcal{S}$ in $\mathbb{R}^3$ as $\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^3 : 2x_1 - x_2 + 3x_3 = 0\}$.

   a) (6 points) Find a **non-orthonormal** basis for the subspace $\mathcal{S}$. Find the projection matrix $\boldsymbol{P}$ onto $\mathcal{S}$ using this basis. Using computer or write a python code to aid your calculation is allowed for this problem.

   b) (9 points) Now find an **orthonormal** basis for the subspace $\mathcal{S}$. Find the projection matrix $\boldsymbol{P}$ onto $\mathcal{S}$ using this basis. All computations need be done by hand and the final numbers in the projection need to be fractions (not floats).

c) (5 points) Calculate the distance of a vector $\boldsymbol{x} = \begin{bmatrix} 1 \\ 5 \\ 1 \end{bmatrix}$ to $\mathcal{S}$ using the projection matrix.

3. In 1936 Ronald Fisher published a famous paper on classification titled "The use of multiple measurements in taxonomic problems." In the paper, Fisher study the problem of classifying iris flowers based on measurements of the sepal and petal widths and lengths, depicted in the image below.



Fisher's dataset (`fisheriris.mat`) is widely available on the web (e.g., Wikipedia). We have also uploaded a copy on Canvas under the HW3 assignment. The dataset consists of 50 examples of three types of iris flowers. The sepal and petal measurements can be used to classify the examples into the three types of flowers.

a) (4 points) Formulate the classification task as a least squares problem. Least squares will produce real-valued predictions, not discrete labels or categories. What might you do to address this issue?

b) (9 points) In class we discussed that when making predictions using least squares, the residuals $\boldsymbol{r}$ are orthogonal to the columns of $\boldsymbol{X}$. Now we will verify this on real data. First implement your solution from part (a) to create a numerical vector $\boldsymbol{y}$ and solve the least squares problem $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$. Compute the residuals, and then make a plot or show some code output to show the residuals are in fact orthogonal to the columns of $\boldsymbol{X}$. Below is some code to help you get started in Python.

```
# STARTER CODE
import numpy as np
import scipy.io
import matplotlib.pyplot as plt
from matplotlib import colors
# load data, make sure 'fisheriris.mat' is in your working directory
```

```
data = scipy.io.loadmat("fisheriris.mat")
X = data['meas']
y_text = data['species']

############################
# YOUR CODE BELOW
# Process and assign numerical values to
# `y` according to your (a), make sure `y` is a 1d numpy array.
# If dimensions are mismatching, you may find `y = y.flatten()` useful.


# Compute the least squares weights


# Compute the residuals


# Make a plot
```

c) (9 points) Write a Python program to train a classifier using LS based on 40 labeled examples of each of the three flower types, and then test the performance of your classifier using the remaining 10 examples from each type. Repeat this with many randomly chosen subsets of training and test. What is the average test error (number of mistakes divided by 30)?

Python starter code:

```
# STARTER CODE
import numpy as np
import scipy.io
# load data, make sure `fisheriris.mat` is in your working directory
data = scipy.io.loadmat("fisheriris.mat")
# training data
X = data['meas']
y_text = data['species']

############################
# YOUR CODE BELOW
# Process and assign numerical values to
# `y` according to your (a), make sure `y` is a 1d numpy array.
# If dimensions are mismatching, you may find `y = y.flatten()` useful.

# number of random trials
N = 10_000
# array to store errors
errs = np.zeros(N)
```

```
# size of training set
num_train = 40

for i in np.arange(N):
    # initialize 0-length arrays for the train and holdout indices. These
    # arrays will be filled in the inner loop.
    idx_train = np.zeros(0, dtype=np.intp)
    idx_holdout = np.zeros(0, dtype=np.intp)

    # There are 3 label types and 50 samples of each type
    for label_type in range(3):
        # Choose a random ordering of the 50 samples
        r = np.random.permutation(50)
        # Add the first num_train indices of the random ordering to
        # the idx_train array
        idx_train = np.concatenate((idx_train,
                                    50 * label_type + r[:num_train]))
        # Add the rest of the indices to the idx_holdout array
        idx_holdout = np.concatenate((idx_holdout,
                                    50 * label_type + r[num_train:]))

    # divide data and labels into the train and holdout sets
    Xt = X[idx_train]
    yt = y[idx_train]
    Xh = X[idx_holdout]
    yh = y[idx_holdout]

    ########################
    # YOUR CODE BELOW
    # Solve for the LS weights


    # Make predictions using the LS weights


    # Turn the real-valued predictions into class labels


    # Compute the errors
```

d) (9 points) Experiment with even smaller-sized training sets. Clearly we need at least one training example from each type of flower. Make a plot of average test error and the average training error as a function of training set size. Comment on the behavior of the train and test errors as a function of training set size. Looking at the plot, what number should we choose for our training set size?

```
# STARTER CODE
import numpy as np
import scipy.io
# load data, make sure ‘fisheriris.mat‘ is in your working directory
data = scipy.io.loadmat("fisheriris.mat")
# training data
X = data['meas']
y_text = data['species']
############################
# YOUR CODE BELOW
# Process and assign numerical values to
# ‘y‘ according to your (a), make sure ‘y‘ is a 1d numpy array.
# If dimensions are mismatching, you may find ‘y = y.flatten()‘ useful.


# number of random trials
N = 1_000
# Min / Max size of the training set
min_num_train = 4
max_num_train = 40

# Arrays to store error rates
train_errs = np.zeros((max_num_train-min_num_train, N))
test_errs = np.zeros((max_num_train-min_num_train, N))

n_train_vals = np.arange(min_num_train, max_num_train)


for j, n_train in enumerate(n_train_vals):
    for i in np.arange(N):
        # initialize 0-length arrays for the train and holdout indices.
        # These arrays will be filled in the inner loop.
        idx_train = np.zeros(0, dtype=np.intp)
        idx_holdout = np.zeros(0, dtype=np.intp)

        # There are 3 label types and 50 samples of each type
        for label_type in range(3):
            # Choose a random ordering of the 50 samples
            r = np.random.permutation(50)
            # Add the first num_train indices of the random ordering to
            # the idx_train array
            idx_train = np.concatenate((idx_train,
                                        50 * label_type + r[:num_train]))
            # Add the rest of the indices to the idx_holdout array
            idx_holdout = np.concatenate((idx_holdout,
```

```
                                        50 * label_type + r[num_train:]))

        # divide data and labels into the train and holdout sets
        Xt = X[idx_train]
        yt = y[idx_train]
        Xh = X[idx_holdout]
        yh = y[idx_holdout]

        ########################
        # YOUR CODE BELOW


###################
# YOUR CODE BELOW
# Make a plot of the train and test errors as a function of
# training set size
```

e) (9 points) Now design a classifier using only the measurements sepal length and petal length. These measurements are in the first and third columns of the matrix `X`. What is the average test error in this case?

4. Subspaces and Bases

a) (6 points) Given a matrix $X = \begin{bmatrix} 0 & 4 & 3 \\ 3 & 6 & 0 \\ 1 & 2 & 0 \end{bmatrix}$, with corresponding projection matrix $P_{\mathcal{S}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & .9 & .3 \\ 0 & .3 & .1 \end{bmatrix}$, find a basis for the column span of X, which is the subspace $\mathcal{S}$.

b) (7 points) $\mathcal{S}$ lives in $\mathbb{R}^3$, so we can write it as an equation $ax_1 + bx_2 + cx_3 = 0$. This equation is true for all vectors $\boldsymbol{x}$ that live in the subspace $\mathcal{S}$. Solve for a, b, and c, and write out the expression for $\mathcal{S}$.

c) (7 points) Use python to find an orthonormal basis for the subspace. Here are some starter code.

```
### STARTER CODE
import numpy as np
import numpy.linalg as la
p = np.array(
    [[0., 4, 3 ],
     [3, 6, 0. ],
     [1. , 2 , 0. ]]
)
```

```
    ### YOUR CODE BELOW
```