Instructions: (Please read carefully and follow them!)
Try to solve all problems on your own. If you have difficulties, ask the instructor or TAs.

Please follow the instructions given below to prepare your solution notebooks:

- Please use different notebooks for solving different Exercise problems.
- The notebook name for Exercise 1 should be `YOURROLLNUMBER_IE684_Lab9_Ex1.ipynb`.
- Similarly, the notebook name for Exercise 2 should be `YOURROLLNUMBER_IE684_Lab9_Ex2.ipynb`, etc.
- Please post your doubts in MS Teams Discussion Forum channel so that TAs can clarify.

There are only 3 exercises in this lab. Try to solve all problems on your own. If you have difficulties, ask the Instructors or TAs.

Only the questions marked [**R**] need to be answered in the notebook. You can either print the answers using `print` command in your code or you can write the text in a separate text tab. To add text in your notebook, click `+Text`. Some questions require you to provide proper explanations; for such questions, write proper explanations in a text tab. Some questions require the answers to be written in LaTeX notation. Please see the demo video (posted in Lab 1) to know how to write LaTeX in Google notebooks. Some questions require plotting certain graphs. Please make sure that the plots are present in the submitted notebooks. Please include all answers in your `.pynb` files.

After completing this lab's exercises, click File → Download .ipynb and save your files to your local laptop/desktop. Create a folder with name `YOURROLLNUMBER_IE684_Lab9` and copy your `.ipynb` files to the folder. Then zip the folder to create `YOURROLLNUMBER_IE684_Lab9.zip`. Then upload only the `.zip` file to Moodle. There will be extra marks for students who follow the proper naming conventions in their submissions.

There will be extra marks for students who follow the proper naming conventions in their submissions.

Please check the **submission deadline announced in moodle.**

Binary Classification Problem

In the last lab, we developed an optimization method to solve the optimization problem associated with binary classification problem. In this lab, we will introduce constraints to the optimization problem and try to extend the scheme we developed in the last lab.
Recall that for a data set $D = \{(x^i, y^i)\}_{i=1}^n$ where $x^i \in \mathscr{X} \subseteq \mathbb{R}^d$, $y^i \in \{+1, -1\}$, we solve:

$$\min_{w \in \mathbb{R}^d} f(w) = \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} L(y^i, w^\top x^i). \tag{1}$$

where we considered the following loss functions:

- $L_h(y^i, w^\top x^i) = \max\{0, 1 - y^i w^\top x^i\}$ (hinge)
- $L_\ell(y^i, w^\top x^i) = \log(1 + \exp(-y^i w^\top x^i))$ (logistic)
- $L_{sh}(y^i, w^\top x^i) = (\max\{0, 1 - y^i w^\top x^i\})^2$. (squared hinge)

Solving the optimization problem (1) facilitates in learning a classification rule $h : \mathscr{X} \to \{+1, -1\}$. We used the following prediction rule for a test sample $\hat{x}$:

$$h(\hat{x}) = \text{sign}(w^\top \hat{x}). \tag{2}$$

In the last lab, we used a decomposition of $f(w)$ and solved an equivalent problem of the following form:

$$\min_w f(w) = \min_w \sum_{i=1}^{n} f_i(w). \tag{3}$$

In this lab, we will consider a constrained variant of the optimization problem (1).
For a data set $D = \{(x^i, y^i)\}_{i=1}^n$ where $x^i \in \mathscr{X} \subseteq \mathbb{R}^d$, $y^i \in \{+1, -1\}$, we solve:

$$\min_w f(w) = \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} L(y^i, w^\top x^i), \text{ s.t. } w \in \mathcal{C} \tag{4}$$

where $\mathcal{C} \subset \mathbb{R}^d$ is a closed convex set.
Hence we would develop optimization method to solve the following equivalent constrained problem of (4):

$$\min_{w \in \mathcal{C}} f(w) = \min_{w \in \mathcal{C}} \sum_{i=1}^{n} f_i(w). \tag{5}$$

Exercise 0: Data Preparation

1. Use the following code snippet. Load the wine dataset from `scikit-learn` package using the following code. We will load the features into the matrix A such that the i-th row of A will contain the features of i-th sample. The label vector will be loaded into y.
   (a) **[R]** Check the number of classes C and the class label values in wine data. Check if the class labels are from the set $\{0, 1, \ldots, C-1\}$ or if they are from the set $\{1, 2, \ldots, C\}$.
   (b) When loading the labels into y do the following:
   - If the class labels are from the set $\{0, 1, \ldots, C-1\}$ convert classes $0, 2, 3, \ldots, C-1$ to -1.
   - If the class labels are from the set $\{1, 2, \ldots, C\}$ convert classes $2, 3, \ldots, C$ to -1.

   Thus, you will have class labels eventually belonging to the set $\{+1, -1\}$.
2. Normalize the columns of A matrix such that each column has entries in range $[-1, 1]$.
3. Note that a shuffled index array `indexarr` is used in the code. Use this index array to partition the data and labels into train and test splits. In particular, use the first 80% of the indices to create the training data and labels. Use the remaining 20% to create the test data and labels. Store them in the variables `train_data`, `train_label`, `test_data`, `test_label`.

```
import numpy as np
#we will load the wine data from scikit-learn package
from sklearn.datasets import load_wine
wine = load_wine()
#check the shape of wine data
print(wine.data.shape)
A = wine.data
#Normalize columns of A so that all entries are in the range [-1,+1]
#for i in range(A.shape[1]):
#  A[i] = ???
#check the shape of wine target
print(wine.target.shape)
#How many labels does wine data have?
#C=num_of_classes
#print(C)
n = wine.data.shape[0] #Number of data points
d = wine.data.shape[1] #Dimension of data points

#In the following code, we create a nx1 vector of target labels
y = 1.0*np.ones([A.shape[0],])
for i in range(wine.target.shape[0]):
   # y[i] = ???? # Convert class labels that are not 1 into -1

#Create an index array
indexarr = np.arange(n) #index array
np.random.shuffle(indexarr) #shuffle the indices
#print(indexarr) #check indexarr after shuffling

#Use the first 80% of indexarr to create the train data and the remaining 20% to
    create the test data
#train_data = ????
#train_label = ????
#test_data = ????
#test_label = ????
```

4. Use the python function developed in last lab where you had implemented the prediction rule in eqn. (2).

5. Use the python function developed in the previous lab which takes as input the model parameter w, data features and labels and returns the accuracy on the data.

Exercise 2: An Optimization Algorithm

1. To solve the problem (5), we shall use the following method (denoted by **ALG-LAB9**). Assume that the training data contains n_{train} samples.

   (a) For $t = 1, 2, 3, \ldots$, do:

      i. Sample i uniformly at random from $\{1, 2, \ldots, n_{train}\}$.
      ii. $w^{t+1} = \text{Proj}_{\mathcal{C}}(w^t - \eta_t \nabla f_i(w^t))$.

   The notation $\text{Proj}_{\mathcal{C}}(z) = \arg\min_{u \in \mathcal{C}} \|u - z\|_2$ denotes the orthogonal projection of point z onto set $\mathcal{C}$. In other words, we wish to find a point $u^\star \in \mathcal{C}$ which is closest to z in terms of ℓ_2 distance. For specific examples of set $\mathcal{C}$, the orthogonal projection has a nice closed form.

2. When $\mathcal{C} = \{w \in \mathbb{R}^d : \|w\|_\infty \leq 1\}$, find an expression for $\text{Proj}_{\mathcal{C}}(z)$. (**Recall**: For a $w = \begin{bmatrix} w_1 & w_2 & \ldots & w_d \end{bmatrix}^\top \in \mathbb{R}^d$, we have $\|w\|_\infty = \max\{|w_1|, |w_2|, \ldots, |w_d|\}$.)

3. Consider the hinge loss function L_h. Use the python modules developed in the last lab to compute the loss function L_h, and objective function value. Also use the modules developed in the last lab to compute the gradient (or sub-gradient) of $f_i(w)$ for the loss function L_h. Denote the (sub-)gradient by $g_i(w) = \nabla_w f_i(w)$.

4. Define a module to compute the orthogonal projection onto set $\mathcal{C}$.

```
def compute_orthogonal_projection(z):
  #complete the code
```

5. Modify the code developed in the previous lab to implement ALG-LAB8. Use the following template.

```
def OPT1(data,label,lambda, num_epochs):
  t = 1
  #initialize w
  #w = ???
  arr = np.arange(data.shape[0])
  for epoch in range(num_epochs):
    np.random.shuffle(arr) #shuffle every epoch
    for i in np.nditer(arr): #Pass through the data points
      # step = ???
      # Update w using w <- Proj_C(w - step * g_i (w))
      t = t+1
      if t>1e4:
        t = 1
  return w
```

6. In OPT1, use `num_epochs`=500, `step`=$\frac{1}{t}$. For each $\lambda \in \{10^{-3}, 10^{-2}, 0.1, 1, 10\}$, perform the following tasks:

   (a) [**R**] Plot the objective function value in every epoch. Use different colors for different λ values.

   (b) [**R**] Plot the test set accuracy in every epoch. Use different colors for different λ values.

   (c) [**R**] Plot the train set accuracy in every epoch. Use different colors for different λ values.

   (d) [**R**] Tabulate the final test set accuracy and train set accuracy for each λ value.

(e) [**R**] Explain your observations.

7. [**R**] Repeat the experiments (with `num_epochs`=500) for different loss functions L_ℓ and L_{sh}. Explain your observations.

Exercise 3: A different constraint set

1. When $\mathcal{C} = \{w \in \mathbb{R}^d : \|w\|_1 \leq 1\}$, find an expression for $\text{Proj}_{\mathcal{C}}(z)$. (**Recall**: For a $w = \begin{bmatrix} w_1 & w_2 & \dots & w_d \end{bmatrix}^\top \in \mathbb{R}^d$, we have $\|w\|_1 = \sum_{i=1}^d |w_i|$.)
2. Consider the hinge loss function L_h. Use the python modules developed in the last lab to compute the loss function L_h, and objective function value. Also use the modules developed in the last lab to compute the gradient (or sub-gradient) of $f_i(w)$ for the loss function L_h. Denote the (sub-)gradient by $g_i(w) = \nabla_w f_i(w)$.
3. Define a module to compute the orthogonal projection onto set $\mathcal{C}$.

```
def compute_orthogonal_projection_ex3(z):
  #complete the code
```

4. In OPT1, use `num_epochs`=500, `step`=$\frac{1}{t}$. For each $\lambda \in \{10^{-3}, 10^{-2}, 0.1, 1, 10\}$, perform the following tasks:
   (a) [**R**] Plot the objective function value in every epoch. Use different colors for different λ values.
   (b) [**R**] Plot the test set accuracy in every epoch. Use different colors for different λ values.
   (c) [**R**] Plot the train set accuracy in every epoch. Use different colors for different λ values.
   (d) [**R**] Tabulate the final test set accuracy and train set accuracy for each λ value.
   (e) [**R**] Explain your observations.
5. [**R**] Repeat the experiments (with `num_epochs`=500) for different loss functions L_ℓ and L_{sh}. Explain your observations.