Instructions: (Please read carefully and follow them!)

Try to solve all problems on your own. If you have difficulties, ask the instructor or TAs.

In this session, we will shift the theme of decomposing a problem with optimization procedures to handle large data to Classification problems.

The implementation of the optimization algorithms in this lab will involve extensive use of the `numpy` Python package. It would be useful for you to get to know some of the functionalities of `numpy` package. For details on `numpy` Python package, please consult https://numpy.org/doc/stable/index.html

For plotting purposes, please use `matplotlib.pyplot` package. You can find examples in the site https://matplotlib.org/examples/.

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_Lab07_Ex1.ipynb`.
- Similarly, the notebook name for Exercise 2 should be `YOURROLLNUMBER_IE684_Lab07_Ex2.ipynb`, etc and so on.

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

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. (**Write the comments and observations with appropriate equations in LaTeX only.**) Some questions require plotting certain graphs. Please make sure that the plots are present in the submitted notebooks.

After completing this lab's exercises, click File → Download .ipynb and save your files to your local laptop/desktop. Create a folder with the name `YOURROLLNUMBER_IE684_Lab07` and copy your `.ipynb` files to the folder. Then zip the folder to create `YOURROLLNUMBER_IE684_Lab07.zip`. Then upload only the `.zip` file to Moodle. **There will be some penalty for students who do not follow the proper naming conventions in their submissions**.

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

In the last lab, you might have recognized that decomposing a problem might help in coming up with optimization procedures to handle large data. In this lab, we will continue with this theme and try to develop procedures that are scalable and achieve reasonably accurate solutions.

Here, we will consider a different problem, namely the binary (or two-class) classification problem in machine learning. The problem is of the following form. For a data set $D = \{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathcal{X} \subseteq \mathbb{R}^d$, $y_i \in \{+1, -1\}$, we solve:

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

Note that we intend to learn a classification rule $h : \mathcal{X} \to \{+1, -1\}$ by solving the problem (1). We will use the following prediction rule for a test sample $\hat{x}$:

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

We will consider the following loss functions:

- $L_h(y_i, w^T x_i) = \max\{0, 1 - y_i w^T x_i\}$ (hinge)
- $L_l(y_i, w^T x_i) = \log(1 + \exp(-y_i w^T x_i))$ (logistic)
- $L_{sh}(y_i, w^T x_i) = (\max\{0, 1 - y_i w^T x_i\})^2$ (squared hinge)

Exercise 1 *For an example $(x, y) \in \mathcal{X} \times \mathcal{Y}$, assume $z = yw^T x$. Then note that the loss functions $L_h, L_l and L_{sh}$ can be equivalently written as $G_h(z), G_l(z), G_{sh}(z)$. Write the loss functions $G_h(z), G_l(z), G_{sh}(z)$ as the functions of z. Plot these loss functions $G_h(z), G_l(z)$ and $G_{sh}(z)$ where z takes the values on the real line $[-\infty, \infty]$. Distinguish the loss functions using different colors. Comment on the behavior of respective loss functions with respect to z.*

Exercise 2 ***Data Preparation***

Use the following code snippet. Load the iris dataset from the `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.*

1. *Check the number of classes C and the class label values in iris data. Check if the class labels are set from the set $\{0, 1, \ldots, C-1\}$ or if they are from the set $\{1, 2, \ldots C\}$.*
2. *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 $\{+1, -1\}$
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 iris data from scikit-learn package
from sklearn.datasets import load_iris
iris = load_iris()
#check the shape of iris data
print(iris.data.shape)

A = iris.data
#check the shape of iris target
print(iris.target.shape)
#How many labels does iris data have?
#C=num_of_classes
#print(C)

n = iris.data.shape[0] #Number of data points
d = iris.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(iris.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. *Write a python function that implements the prediction rule in eqn. (2). Use the following the code template.*

```
def predict(w,x):
 #return ???
```

5. *Write a Python function that takes as input the model parameter w, data features, and labels and returns the accuracy of the data. (Use the predict function).*

```
def compute_accuracy(data, labels, model_w):
  #Use predict function defined above
  #return ???
```

Exercise 3 ***An Optimization Algorithm***

1. *Note that problem (1) can be written as*

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

   Find an appropriate choice of $f_i(w)$.

2. *Consider the loss function L_h. Write a Python module to compute the loss function L_h.*

3. *Write a Python routine to compute the objective function value. You can use the function used for computing the loss.*

4. *Write an expression to compute the gradient (or sub-gradient) of fi(w) for the loss function L_h. Denote the (sub-)gradient by $g_i(w) = \nabla_w f_i(w)$. Define a python function to compute the gradient.*

5. *Write an optimization algorithm where you pass through the training samples one by one and do the (sub-)gradient updates for each sample. Recall that this is similar to ALG-LAB7. 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 <- w - step * g_i (w)
t = t+1
if t>1e4:
t = 1
return w
```

6. *in OPT1, use* `num_epochs` *= 1000,* `step` $= \frac{1}{t}$*. For each $\lambda \in \{10^{-3}, 10^{-2}, 0.1, 1, 10\}$, perform the following tasks:*

   - *Plot the objective function value in every epoch. Use different colors for different λ values.*
   - *Plot the test set accuracy in every epoch. Use different colors for different λ values.*
   - *Plot the train set accuracy in every epoch. Use different colors for different λ values.*
   - *Tabulate the final test set accuracy and train set accuracy for each λ value.*
   - *Explain your observations.*

7. *Note that in OPT1, a fixed number of epochs is used. Can you think of some other suitable stopping criterion for terminating OPT1? Implement your stopping criterion and check how it differs from the one in OPT1. Use* `step` $= \frac{1}{t}$ *and λ which achieved the best test set accuracy in the previous experiment.*

8. *Repeat the experiments (with* `num_epochs`*=1000 and with your modified stopping criterion) for different loss functions L_l and L_{sh}. Explain your observations*