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_Lab08_Ex1.ipynb`.
- Similarly, the notebook name for Exercise 2 should be `YOURROLLNUMBER_IE684_Lab08_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_Lab08` and copy your `.ipynb` files to the folder. Then zip the folder to create `YOURROLLNUMBER_IE684_Lab08.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, we developed an optimization method to solve the optimization problem associated with binary classification problems. Recall that for a dataset $D = \{(x^i, y^i)\}_{i=1}^n$ where $x^i \in \mathcal{X} \in \mathbb{R}^d$, $y^i \in \{+1, -1\}$, we solve:

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

where we considered 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)

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

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

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 dataset $D = \{(x^i, y^i)\}_{i=1}^n$ where $x^i \in \mathcal{X} \in \mathbb{R}^d$, $y^i \in \{+1, -1\}$, we solve:

$$\min_{w \in \mathbb{R}^d} f(w) = \frac{\lambda}{2}||w||_2^2 + \frac{1}{n}\sum_{i=1}^{n} L(y_i, w^T x_i), \text{ s.t. } w \in \mathcal{C} \tag{4}$$

where $\mathcal{C} \in \mathbb{R}^d$ is a closed convex set.

Hence we would develop an optimization method to solve the following equivalent constrained problem of (4):

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

Let's start coding now!

Exercise 1 ***(Data Preparation)*** *Load the wine 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 wine 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. *Normalize the columns of A matrix such that each columns has entries in range $[-1, 1]$.*
4. *Create an index array of size number of samples. 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`.
5. *Write a Python function that implements the prediction rule.*
6. *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).*

Exercise 2 ***An Optimization Algorithm***

1. *To solve the problem (5), we shall use the following method (denoted by ALG-LAB8). Assume that the training data contains n_{train} samples.*

   For $t = 1, 2, 3, \dots,$ do:

   (a) *Sample i uniformly at random from $\{1, 2, \dots, n_{train}\}$*
   (b) $w^{t+1} = Proj_{\mathcal{C}}(w^t - \eta_t \nabla f_i(w^t))$.

   The notation $Proj_{\mathcal{C}} = \arg\min_{u \in \mathcal{C}} ||u - z||_2$ denotes the orthogonal projection of point z onto set C. In other words, we wish to find a point $u^* \in \mathcal{C}$ 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 and expression for $Proj_{\mathcal{C}}(z)$. (**Recall**: For a $w = [w_1 w_2 \dots w_d]^T \in \mathbb{R}^d$, we have $||w||_\infty = \max\{|w_1|, |w_2|, \dots, |w_d|\}$.)*

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 the set $\mathcal{C}$.*

5. *Modify the code template given in the last lab to implement ALG-Lab8. Use the following code 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` = 500, `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. *Repeat the experiments (with `num_epochs`=500 and with your modified stopping criterion) 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* $Proj_{\mathcal{C}}(z)$. *(**Recall:** For a* $w = [w_1 w_2 \dots w_d]^T \in \mathbb{R}^d$, *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* $\mathcal{C}$.
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:*
   - *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.*
5. *Repeat the experiments (with* `num_epochs`*=500 and with your modified stopping criterion) for different loss functions* L_l *and* L_{sh}. *Explain your observations*