Lab#3: Logistic Regression

Prepared by: Teeradaj Racharak (r.teeradaj@gmail.com)

Scenario 1: Consider the data $(\boldsymbol{X}, y)$ given in the cell below.

```
X1 = np.array([[ 5.85811186e+00,  1.22514944e-01], [ 6.00000443e+00,  1.35309858e+00],
       [ 5.92284784e+00,  2.73055315e+00], [ 1.52096299e+00,  6.21018604e+00],
       [ 5.30181969e+00,  1.06006684e+00], [ 1.84827940e-02,  2.85281664e+00],
       [ 3.45604657e+00,  1.53032968e+00], [ 4.22908122e-03,  3.36294834e+00],
       [ 3.48985476e+00,  7.51884752e-01], [ 6.26725305e+00,  1.07496703e+00],
       [ 1.14301185e+00,  6.51237438e+00], [ 4.23216371e+00, -2.44093332e+00],
       [ 5.47550663e+00,  4.06333297e-01], [ 6.91545908e-01,  6.40202254e+00],
       [ 5.36323789e+00,  3.34195653e-01], [ 2.05562456e+00,  6.22292165e+00],
       [ 8.41550344e-01,  6.64205417e+00], [ 3.52365485e+00,  2.04313164e+00],
       [ 4.47711189e+00, -3.86246764e-02], [ 5.89528259e+00,  1.79006628e+00],
       [ 6.19166029e-01,  5.11662188e+00], [ 5.54388098e+00,  1.06447058e+00],
       [ 2.32147724e+00,  5.38777768e+00], [ 2.62141168e+00,  4.28575143e+00],
       [ 4.63535586e-02,  4.45755669e+00], [ 5.33931497e+00,  7.04464916e-01],
       [ 5.57510725e+00, -3.39830880e-02], [ 1.96161624e+00,  5.95425951e+00],
       [ 1.91163042e+00,  6.75226744e+00], [ 1.64362576e+00,  8.10340980e+00],
       [ 7.10795973e-01,  8.76085464e+00], [ 6.92524153e-01,  5.61415508e+00],
       [ 4.94898675e-01,  4.54839212e+00], [ 6.06103845e+00,  2.50633947e+00],
       [ 9.73818272e-02,  2.98938521e+00], [ 4.88362231e+00, -1.27782382e+00],
       [ 3.07408705e+00,  3.63515130e+00], [ 3.28864750e+00,  2.07462075e+00],
       [ 6.26354557e+00,  4.53934384e+00], [ 6.16816836e+00,  2.73690942e+00],
       [ 4.21560198e+00, -1.22336763e+00], [ 5.12756678e+00, -1.29691280e+00],
       [ 4.35537106e+00, -1.02344862e+00], [ 2.45336946e+00,  5.36145405e+00],
       [ 1.81532920e+00,  7.15020457e+00], [ 5.07147058e+00,  1.02288646e-01],
       [ 1.54560026e+00,  7.04755026e+00], [ 4.42717299e+00, -2.81259623e-01],
       [ 2.83503002e+00,  4.84934257e+00], [ 5.80106290e+00, -5.74634830e-02],
       [ 2.22759281e+00,  5.29488901e+00], [ 3.08219947e+00,  5.00507772e+00],
       [ 5.79711865e+00,  5.07300906e-01], [ 1.50372185e+00,  7.92150886e+00],
       [ 5.31174440e+00,  4.69530673e-01], [ 1.02455161e+00,  4.37343480e+00],
       [ 4.52702912e+00, -1.04276511e-01], [ 3.88224756e+00, -8.80499022e-01],
       [ 4.05922572e+00, -2.13265308e-01], [ 6.03892811e+00,  2.54764213e+00],
       [ 6.69758084e-01,  8.75729172e+00], [ 3.43081467e+00,  1.78550466e+00],
       [ 5.68295684e+00,  2.56567127e+00], [ 5.77236666e+00,  8.62633177e-01],
       [ 4.62600300e+00, -1.59433537e+00], [ 5.44977295e+00, -6.29484710e-01],
       [ 8.83574711e-01,  6.39890091e+00], [ 9.77493464e-01,  7.04576842e+00],
       [ 6.96010647e-01,  6.32483711e+00], [ 2.42501319e+00,  5.31169779e+00],
       [ 7.13900135e-01,  5.55653866e+00], [ 1.60388408e+00,  6.52536492e+00],
       [ 1.95919971e+00,  6.14051234e+00], [ 3.72469206e+00,  1.16211293e-01],
       [ 2.18109919e+00,  5.03391632e+00], [ 3.37992980e-01,  4.38666016e+00],
       [ 2.66255987e+00,  4.16248548e+00], [ 1.37563677e+00,  9.01655089e+00],
       [ 4.97070262e+00,  7.65748319e-01], [ 5.67177205e+00,  1.60946513e+00],
       [ 5.31404919e+00,  9.70842964e-02], [ 6.02115102e+00,  1.14662909e+00],
       [ 4.05204793e+00, -1.03418242e+00], [ 3.66918406e+00,  1.11617948e+00],
       [ 3.82708241e+00, -6.89595628e-01], [ 4.65955271e+00, -8.63810375e-01],
       [ 3.32919754e+00,  2.31647457e+00], [ 4.33734390e+00, -1.29535742e+00],
       [ 3.95943543e+00,  4.58081833e-01], [ 2.89422338e+00,  4.49606922e+00],
       [ 2.19510893e+00,  7.36292377e+00], [ 2.79133907e+00,  4.25746815e+00],
       [ 7.19221108e-01,  6.54004102e+00], [ 5.13548270e+00, -2.54674801e-01],
       [ 2.05933161e+00,  5.28710542e+00], [ 1.78202945e+00,  7.16853185e+00],
       [ 3.48957680e+00,  1.36969854e+00], [ 4.93781831e+00,  6.91655345e-01],
       [ 4.27909400e+00,  2.03346686e-02], [ 5.15336506e+00, -4.14053935e-01]])
```

```
X2 = np.array([[ 3.43323051, -4.68184123], [ 2.59748185, -0.61500731],
       [ 0.33480371, -1.47508942], [ 5.7825976 , -5.44824068],
       [ 0.87070876, -0.89526139], [ 0.01802905, -1.89299839],
       [ 2.74929135, -0.56312653], [ 4.22560236, -6.12770556],
       [ 2.78127731, -0.95800779], [ 3.47222918, -3.33378672],
       [ 0.9782014 , -1.00938521], [ 2.25445075, -0.90402158],
       [ 5.55216081, -6.18491739], [ 6.13910761, -3.93666989],
       [ 2.99043663, -2.89123684], [ 0.44884867, -1.45297558],
       [ 3.88912523, -5.06048717], [ 2.10025175, -1.1028566 ],
       [ 0.93182831,  1.75229369], [ 3.80362678, -4.04122195],
       [ 0.34639776, -0.87243213], [ 5.7116009 , -4.93209123],
       [ 5.05087165, -6.67367203], [ 1.3492558 ,  1.33394759],
       [ 2.95688016, -2.19674503], [ 4.96859322, -8.01640548],
       [ 5.87695275, -3.34649668], [ 1.94117582,  2.27797699],
       [ 3.62547526, -4.0217949 ], [ 0.82629384, -0.27290357],
       [ 3.6890367 , -5.82061241], [ 4.93797566, -6.52602515],
       [ 5.90841492, -5.27521196], [ 5.1879215 , -7.80427044],
       [ 2.09766098, -0.68048301], [ 5.76655061, -3.82518968],
       [ 0.97690188, -0.91069622], [ 2.26501337,  1.56056043],
       [ 3.13676069, -2.29446312], [ 5.29747136, -5.55385824],
       [ 2.00649857,  0.68266315], [ 0.77856457, -1.75844385],
       [ 3.54571209, -4.98304275], [ 0.97247461, -0.11610218],
       [ 4.31311129, -7.82947521], [ 3.48007352, -6.04086403],
       [ 1.50781401,  0.33381045], [ 2.43853928, -0.03145604],
       [ 0.25314147, -2.45908731], [ 1.35654226, -0.22865539],
       [ 5.06582758, -6.90535573], [ 3.41686466, -3.37239301],
       [ 5.61214807, -5.25701105], [ 5.97522182, -3.22566212],
       [ 3.18939015, -4.3188073 ], [ 5.46345617, -4.82657628],
       [ 5.41151575, -6.0126475 ], [ 5.347017  , -6.83984988],
       [ 5.7259852 , -3.72566258], [ 1.38905244, -0.13844681],
       [ 3.9562558 , -6.09691018], [ 1.25473486,  0.8640623 ],
       [ 0.08936814, -3.30723552], [ 3.22365241, -4.78721191],
       [ 3.2415687 , -4.24880257], [ 1.1302129 , -1.41426559],
       [ 4.12636748, -7.32837589], [ 2.15023019,  0.05432966],
       [ 4.55269902, -6.73020725], [ 1.63594817,  1.72806532],
       [ 0.099722  , -1.90546332], [ 4.61336823, -7.07377162],
       [ 3.55425568, -3.30089928], [ 4.52715899, -7.27897524],
       [ 5.30602493, -7.61995372], [ 5.44738276, -6.9766927 ],
       [ 4.34946148, -7.64515063], [ 4.04771578, -7.19386371],
       [ 1.85701731,  0.39977015], [ 3.04610896, -3.2246929 ],
       [ 2.96501443, -1.32784036], [ 1.15817849,  0.48576348],
       [ 1.11438724,  1.46252362], [ 0.44152831,  0.09032514],
       [ 1.95859258, -1.05902546], [ 2.63644257, -1.85839175],
       [ 1.93538744,  0.77671848], [ 4.12489761, -4.92407274],
       [ 3.93239509, -5.08468575], [ 3.22929697, -4.11585099],
       [ 4.94033586, -4.20813851], [ 2.77167364, -2.60322887],
       [ 2.05535528,  2.02195992], [ 0.97916579,  0.86646976],
       [ 1.84314531,  2.3764651 ], [ 4.92906354, -6.32746025],
       [ 1.72832737,  1.88550113], [ 2.83989839, -3.62635215],
       [ 4.17304198, -7.15506579], [ 0.47159292,  0.28458576]])
```

Problem 1.1 [Python from scratch]: Place the following code into any editor for seeing the relationship between $(\boldsymbol{X}, y)$. Then, answer if the data are linearly separable? Noted that the following code will be not used in Problem 2.2, Problem 2.3, and Problem 2.4.

```
import matplotlib.pyplot as plt

fig = plt.figure()
axis = fig.add_subplot(111)
axis.scatter(X1[:,0], X1[:,1], marker='o')
axis.scatter(X2[:,0], X2[:,1], marker='x')
plt.show()
```

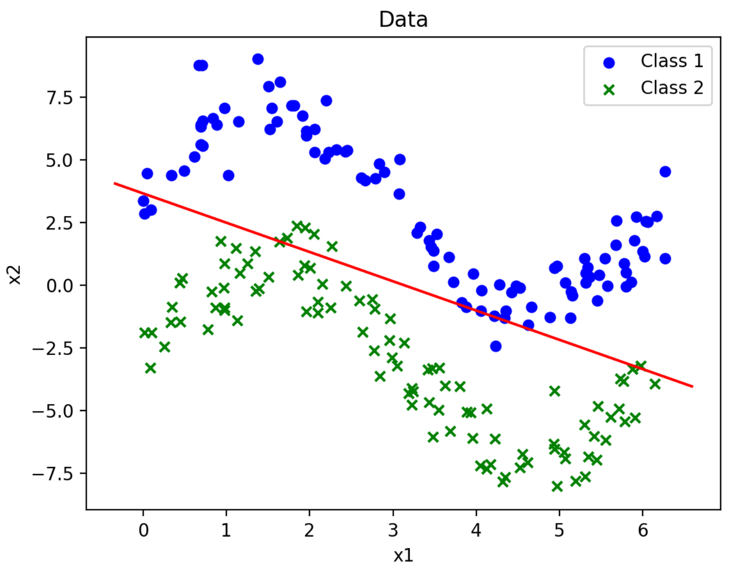
Problem 1.2 [Python with TensorFlow]: Build a binary logistic regression model in TensorFlow to classify $(\boldsymbol{X}, y)$. Train the model for 30,000 epochs using the gradient descent optimizer with 0.03 learning. Answer the best $\boldsymbol{\theta}$ you have obtained.

Problem 1.3 [Python from scratch]: Now, you are familiar with training a binary logistic regression model in TensorFlow. Let's try to make it from scratch using stochastic gradient ascent with a learning rate of 0.01 (see the following pseudocode and the slide as a hint).

```
repeat until converged {
    $\boldsymbol{\theta} := \boldsymbol{\theta} + \alpha \nabla_l(\boldsymbol{\theta})$ # $\alpha := 0.01$
}
```

Depending on the random pattern presentation order, you should reach the optimum after about 3,000 epochs. Answer the best log likelihood and best $\boldsymbol{\theta}$ you obtain.

Problem 1.4 [Python from scratch]: Plot the data with the decision boundary $\boldsymbol{\theta}^T \boldsymbol{x} = 0$. You should be able to obtain a plot as follows:



Scenario 2 (One-vs-all Method): Many machine learning tutorials often refer to MNIST dataset of handwritten digits (this one also refers to !). In this scenario, we will learn to use the one-vs-all method to classify 5,000 images of digits.

The goal is to identify digits from 0 to 9 by looking at 20x20 pixels. We are going to treat each pixel as an individual feature. This means there are 400 features for each image. Noted that the dataset used in this scenario is a portion of the original one, which contains 60,000 examples for training and 10,000 examples for testing.

The dataset is provided via **this link** and some examples of it are displayed as follows.

Usage
In Python, it is easy to import a .mat file using the 'loadmat' function from the 'scipy.io' module and the idea of using it has already provided in the lab guide.

Problem 2.1 [Python with TensorFlow]: Write a program using a logistic regression model for classifying handwritten digits. Recall that the cost function is as follows:

$$J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}\Big[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\Big]$$

Use the gradient descent optimizer with 0.6 learning rate and 5,000 epochs for each classifier. Then, you should obtain around 94.06% accuracy.