Mercury Mcindoe 85594505

1) Logistic Regression

Assume that we have a training set and a test set as follows: Use these data to implement a logistic classifer.

Sample ID	x_1	x_2	y
1	-1.63	-1.36	0
2	-1.67	-1.36	0
3	-1.81	-2.73	0
4	-0.82	-1.40	0
5	-0.71	2.95	1
6	-0.25	1.78	1
7	-0.61	0.79	1
8	0.49	0.56	1

Table 1: Training set

Sample ID	x_1	x_2	y
1	-1.90	-0.95	0
2	-0.27	-0.87	0
3	0.14	-2.23	0
4	0.49	-0.18	1
5	0.97	1.43	1
6	1.07	1.30	1

Table 2: Testing Set

Figure 1: plot for tables

We use the linear model $f_\Theta(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ and the logistic regression function is written as $\sigma_\Theta(x_1, x_2) = \frac{1}{1+e^{-f_\Theta(x_1,x_2)}}$.
We use cross-entropy loss as the loss function. As introduced in the lecture, we will use gradient descent method to update the model based on the data points in the training set. The model parameters are initialized as $\theta_0 = -1, \theta_1 = -1.5, \theta_2 = 0.5$. We set step size $\alpha = 0.1$.

(a) Write down the logistic model $P(\hat{y} = 1|x_1, x_2)$ and its cross-entropy loss function

From the linear model $f_\Theta(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$, we can that the logistic regression function can be expressed as $\sigma_\Theta(x_1, x_2) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\theta_2 x_2)}}$.
Since $P(\hat{y} = 1 \mid x_1, x_2) = \text{logit}^{-1}(X^T\Theta) = \text{logit}^{-1}(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$,
The logistic model would be $P(\hat{y} = 1 \mid x_1, x_2) = \frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\theta_2 x_2)}}$.
$\therefore P(\hat{y} = 1 \mid x_1, x_2) = \frac{1}{1+e^{-(-1-1.5x_1+0.5x_2)}}$
Regarding the cross-entropy function,
$E(\Theta) = -\sum_{i=1}^{8}[\hat{y}^{(i)} \ln\big(P\big(\hat{y}^{(i)} = 1 \mid X^{(i)}\big) + \big(1 - \hat{y}^{(i)}\big) \ln\big(1 - P\big(\hat{y}^{(i)} = 1 \mid X^{(i)}\big)\big)\big]$
$= -\sum_{i=1}^{8}\left[\hat{y}^{(i)} \ln\left(\frac{1}{1+e^{-(-1-1.5x_1+0.5x_2)}}\right) + \big(1 - \hat{y}^{(i)}\big) \ln\left(1 - \frac{1}{1+e^{-(-1-1.5x_1+0.5x_2)}}\right)\right)]$

(b) Use gradient descent to update θ_0, θ_1 and θ_2 for ONE iteration. Write down the equations, gradient values, and updated parameters. Then implement the iterative updating using your own Python algorithm till convergence. The example code is provided in CPEN 355 HW2 Example Codes.ipynb.

We will use the following equations to use gradient descent and update the θ_i values with the initial parameters $\theta_0 = -1, \theta_1 = -1.5, \theta_2 = 0.5$.
$\nabla E(\Theta) = \sum_{i=1}^{m}\left(\frac{1}{1+e^{-(\theta_0+\theta_1 x_1+\theta_2 x_2)}} - y^{(i)}\right)X^{(i)} = \sum_{i=1}^{8}\left(\frac{1}{1+e^{-(-1-1.5x_1+0.5x_2)}} - y^{(i)}\right)X^{(i)}$
The gradient thus become $\nabla E(\Theta) = \begin{pmatrix} 0.50441704 \\ -3.55598493 \\ -6.09547767 \end{pmatrix}$.
Using the above gradient, we can get the updated theta values $\begin{pmatrix} \theta_{0'} \\ \theta_{1'} \\ \theta_2 \end{pmatrix} = \begin{pmatrix} -1 \\ -1.5 \\ 0.5 \end{pmatrix} - 0.1\begin{pmatrix} 0.06305213 \\ -0.44449812 \\ -0.76193471 \end{pmatrix}$

$$= \begin{pmatrix} -1.0504417 \\ -1.14440151 \\ 1.10954777 \end{pmatrix}.$$

However after enough iterations until convergence, we get the values $\begin{pmatrix} \theta_{0'} \\ \theta_{1'} \\ \theta_2 \end{pmatrix} = \begin{pmatrix} 2.40973505 \\ 1.27068101 \\ 6.55841998 \end{pmatrix}$

(c) Visualize the training set and your model's decision hyperplane for your model's state at initialization, after one iteration, and after convergence. For each of the three plots, you may use a scatter plot for visualizing the training data, two different label colors for each class, and a line (derived from Θ of your model) indicating the decision boundary of your model.

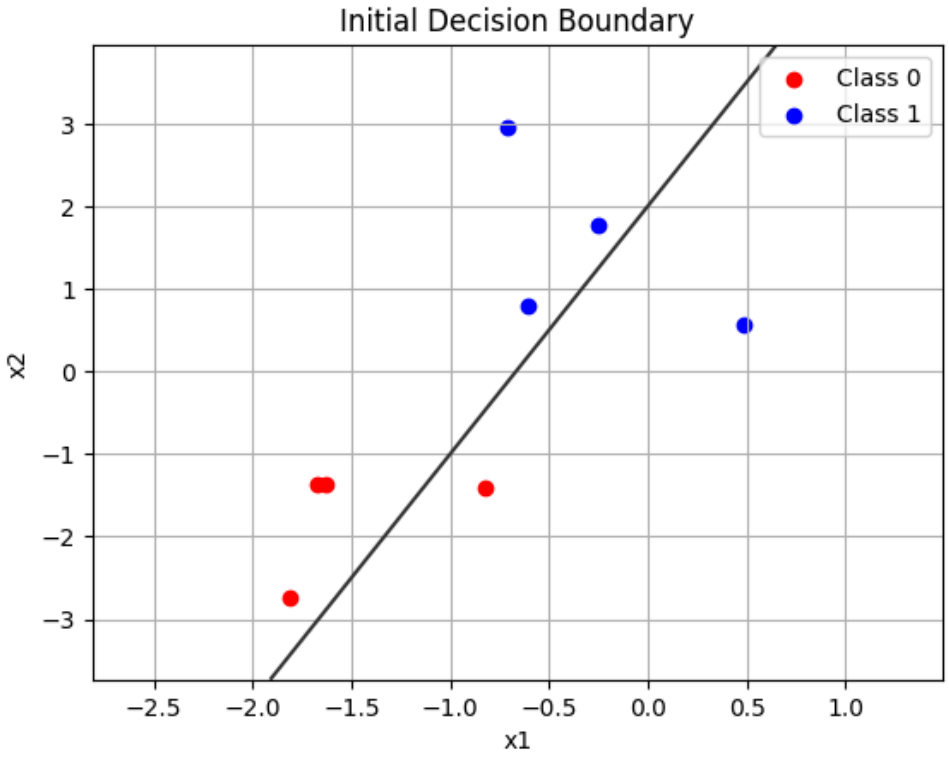

Figure 2: Question 1 tables.

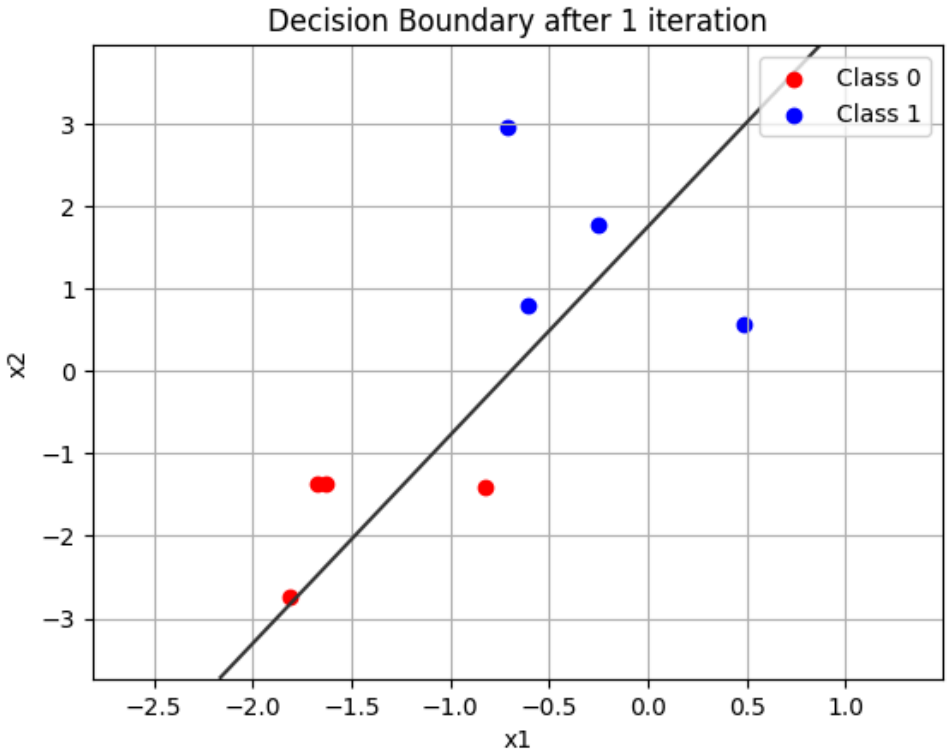

Figure 3: plot for d.

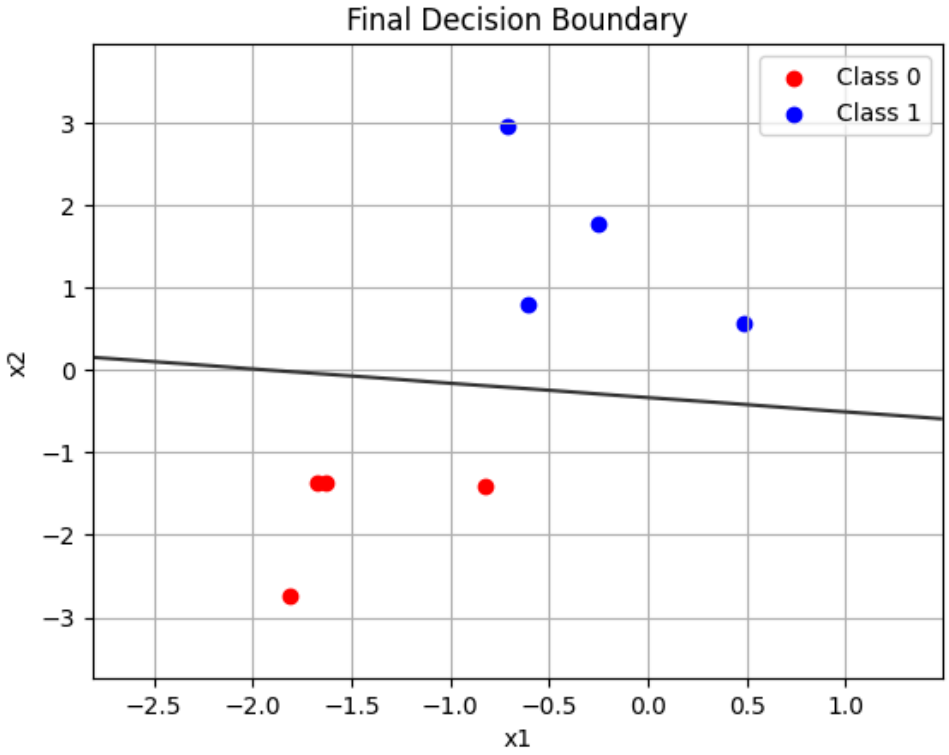

Figure 4: plot for c.

(d) Use Sklearn to find the best θ_0, θ_1 and θ_2 till convergence, and plot the decision boundary of the final parameters. The example code is provided in CPEN 355 HW2 Example Codes.ipynb. You may leave Sklearn's default solver and initialization unchanged to compute the optimal Θ.

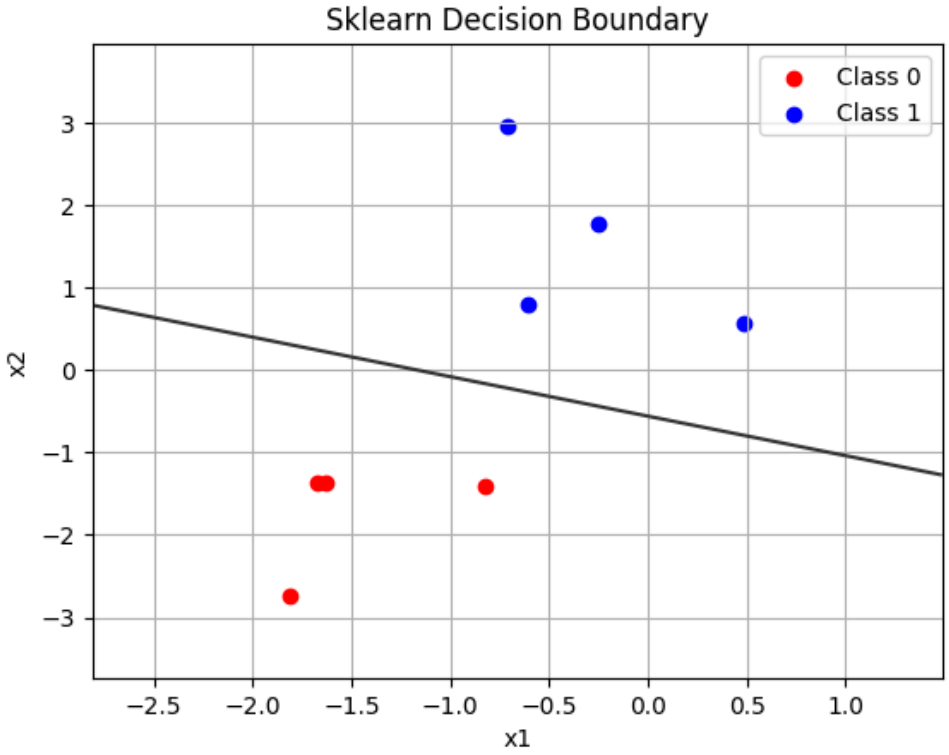

Figure 5: plot for (d).

$$\Theta_{\text{sklearn}} = \begin{pmatrix} 0.65645172 \\ 0.56140902 \\ 1.17249855 \end{pmatrix}$$

(e) Use the above new model to make predictions for all the samples in the test dataset. Compare and verify your final results in (b) and the results generated by Sklearn in (d). Then, using your own calculations/implementations, compute the accuracy, precision, and recall of both models.

Sample ID	$y_{\{\text{pred}\}}$ **by model(b)**	$y_{\{\text{pred}\}}$ **by model(d)**
1	0	0
2	0	0
3	0	0
4	1	1
5	1	1
6	1	1

We will calculate the accuracy, precision and recall of both models.

For our model we know that $\Theta_{\text{final}} = \begin{pmatrix} 1.39551551 \\ 0.68990455 \\ 4.92061964 \end{pmatrix}$ and $\Theta_{\text{sklearn}} = \begin{pmatrix} 0.65645172 \\ 0.56140902 \\ 1.17249855 \end{pmatrix}$.

```
def createTPTNFPFN(y_test, y_pred) :
    dictionary = {
      "TP" : 0,
      "TN" : 0,
      "FP" : 0,
      "FN" : 0
    }
    for i in range(len(y_pred)) :
      if y_pred[i] == 1:
        if y_test[i] == 0 :
          dictionary["FN"] += 1
        else :
          dictionary["TP"] += 1
      else :
        if y_test[i] == 0 :
          dictionary["TN"] += 1
        else :
          dictionary["FP"] += 1

    return dictionary
```

Along with the testing data set, we will use the above python code to calculate accuracy and recall.

Also, we will calculate the y_test data via the following code,

```python
def sigmoid(z) :
  return 1/(1 + np.exp(-z))

theta_final = [1.91250208, 0.99419732, 5.72407974]
probabilities = sigmoid(np.dot(X_test, theta_final))
y_pred = (probabilities >= 0.5).astype(int)

theta_sklearn = np.hstack([model.intercept_, model.coef_.flatten()])

prob_sk = sigmoid(np.dot(X_test, theta_sklearn))
y_pred_sk = (prob_sk >= 0.5).astype(int)
```

Which results in the following results $y_{\text{pred}} = [0, 0, 0, 1, 1, 1], y_{\text{pred\_sk}} = [0, 0, 0, 1, 1, 1]$.
Since we know that $y_{\text{test}} = [0, 0, 0, 1, 1, 1]$, then we can calculate the values for accuracy, precision and recall.
For Θ_{final} from (b)

$$\textbf{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{3+3}{3+3+0+0} = 1.0$$

$$\textbf{precision} = \frac{TP}{TP + FP} = \frac{3}{3+0} = 1.0$$

$$\textbf{recall} = \frac{TP}{TP + FN} = \frac{3}{3+0} = 1.0$$

Similarly, for Θ_{sklearn} from (d) we get,

$$\textbf{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{3+3}{3+3+0+0} = 1.0$$

$$\textbf{precision} = \frac{TP}{TP + FP} = \frac{3}{3+0} = 1.0$$

$$\textbf{recall} = \frac{TP}{TP + FN} = \frac{3}{3+0} = 1.0$$

2) Short Answer Questions

Question 1: In a binary classification problem with Logistic Regression, what is the primary purpose of the decision boundary?
(A) It separates the training data into two clusters.
(B) It defines the line where the predicted probability equals 0.5.
(C) It is a hyperplane that maximizes the margin between classes.
(D) It represents the line with the highest gradient of log-odds.

Question 2: For linear separable data, Logistic Regression can find parameters that achieve exact zero loss on the training set?
(A) Yes
(B) No
(C) Depends