**Machine Learning – HW 3**

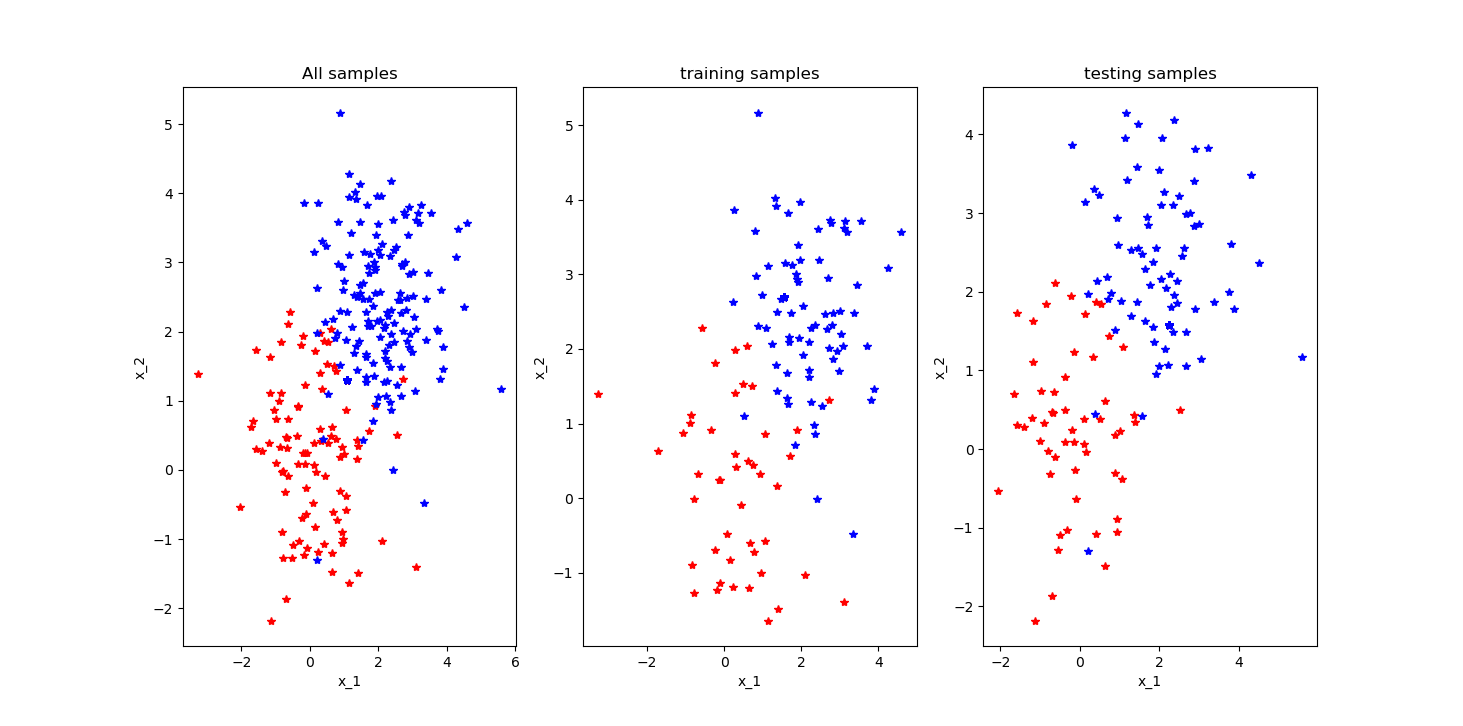
Atieh Kashani

**Problem 1. Logistic Regression:**

In this problem the binary class label for a given data is predicted using Logistic Regression method.

**First step**: In the first step of the code, the generated data is splitted into training and testing subsets. Generated dataset includes both positive and negative samples, and each sample is described with two features. They are totally 250 samples. I have implemented a random permutation, to split the data randomly. The code randomly picks up 120 number of samples for training and use the rest (130) of testing.

The following figures show the splitting results:



**Second step:** for training a logistic regression model using the training data, I have tried the two functions in the provided code in “codeLogic” folder. There are two implementations for training the model: The first one implements the gradient descent (GD) method andanother implementation calls the sklearn library. Following are the results from these two implementations and their performance difference:

* Calculate theta and cost for gradient descent method:

theta [array([0.03623813]), array([0.0513442])]

cost is [0.65542105]

theta [array([0.86769]), array([0.8925124])]

cost is [0.40852937]

theta [array([0.89953753]), array([0.89998225])]

cost is [0.40844691]

theta [array([0.90175569]), array([0.90031955])]

cost is [0.40844653]

theta [array([0.90191161]), array([0.90034086])]

cost is [0.40844653]

* Score over testing samples from sklearn model (Score function applies model over testing samples to make prediction):

0.9153846153846154

* Comparing the accuracy of two models, self-developed model using gradient descent and Sklearn model scores:

Scikit won.. :(

Your score: 0.8076923076923077

Scikits score: 0.9153846153846154

A larger score indicates a better fit.

**Third Step:** I applied the learned model to get the binary classes of testing samples for each of the both implementations in the second step and implemented the related codes. For self-developed model, for getting binary class of testing sample after applying sigmoid function we need a threshold value. I used 0.5 as threshold. So, all probabilities ≥ 0.5 = class 1 and all probabilities < 0 = class 0. For sklearn implementation, I used predict() function from the Sklearn library to get the binary classes of testing samples. Followings are the results:

The (predicted labels) binary classes of testing samples \_ from the self-developed model:

[1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 1. 1. 1.

1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 0.

1. 1. 0. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1.

1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 0. 1. 1.

1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1.

1. 1. 1. 1. 0. 1. 1. 1. 1. 1.]

The ( predicted labels ) binary classes of testing samples \_ from sklearn:

[0. 1. 1. 1. 1. 0. 0. 1. 1. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 1. 1. 1.

1. 0. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 0. 0.

1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1.

1. 1. 1. 1. 0. 0. 1. 0. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 1. 1. 1. 0. 0. 1.

0. 1. 1. 0. 1. 0. 1. 0. 0. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1.

1. 1. 1. 1. 0. 1. 1. 1. 0. 1.]

**Fourth Step:** Compare the predications with the ground-truth label. Calculate average errors and standard deviation:

average error \_ self-developed function: 0.39822485207100594 (0.48953224537718804)

average error \_ Sklearn function: 0.4502958579881657 (0.4975233645436838)

**Problem 2. Confusion Matrix**

**Prediction**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Cat** | **Dog** | **Monkey** |
| **Cat** | **1** | **3** | **1** |
| **Dog**  **Ground-truth** | **3** | **3** | **2** |
| **Monkey** | **2** | **2** | **3** |

Accuracy: Calculated as the sum of correct classifications divided by the total number of classifications

TP(True Positive) / total = (1+3+3) / 20 = 7 / 20 = 0.35

Precision rate for each class:

Precision Class Cat: 1 / (1+3+2) = 1/ 6 = 0.1666

Precision Class Dog: 3 / (3+3+2) = 3/8 = 0.375

Precision Class Monkey: 3/ (1+2+3) = 3/6 = 0.5

Recall rate for each class:

Recall Class Cat: 1/ (1+3+1) = 1/5= 0.2

Recall Class Dog: 3/ (3+3+2) = 3/8= 0.375

Recall Class Monkey: 3/ (2+2+3) = 3/7= 0.4285

**Problem 3. Comparative Studies**

For this problem a function (func\_calCanfusionMatrix ) has been implemented to calculate the confusion matrix for the prediction results of a classifier. This function is implemented in the step 1 of the script “main\_part1.py” and the input values are predY (vector of predicted labels) and trueY (the vector of true labels). This function also calculates the accuracy, class precision of each class and recall rate of each class. Following are the results of calculation the confusion matrix, accuracy, per-class precision and per-class recall rate of both logistic regression implementations:

Calculate confusion matrix for self-developed model of Logistic Regression

[[23, 32], [1, 74]]

Accuracy: 0.7461538461538462

precision\_class\_1: 0.6981132075471698

precision\_class\_0: 0.9583333333333334

recall\_class\_1: 0.9866666666666667

recall\_class\_0: 0.41818181818181815

Calculate confusion matrix for Sklearn model of Logistic Regression

[[50, 5], [3, 72]]

Accuracy: 0.9384615384615385

precision\_class\_1: 0.935064935064935

precision\_class\_0: 0.9433962264150944

recall\_class\_1: 0.96

recall\_class\_0: 0.9090909090909091

According to the results accuracy of the Sklearn implementation of logistic regression model is better.