NYCU Introduction to Machine Learning, Homework 2

109550198, 卜銳凱

**Part. 1, Coding (50%)**:

In this coding assignment, you are requested to implement Logistic Regression and Fisher's Linear Discriminant by using only Numpy. After that, train your model on the provided dataset and evaluate the performance on the testing data.

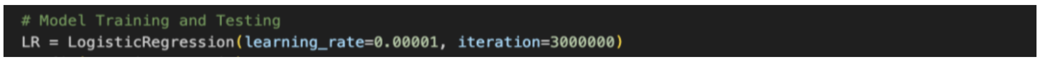
**(15%) Logistic Regression**

**Requirements:**

- Use Gradient Descent to update your model.
- Use CE (Cross-Entropy) as your loss function.
-

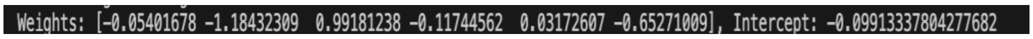
**Criteria:**

1. (0%) Show the hyperparameters (learning rate and iteration) that you used.



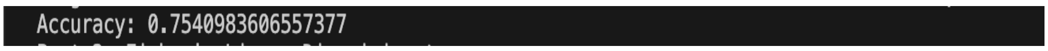
```
# Model Training and Testing
LR = LogisticRegression(learning_rate=0.00001, iteration=3000000)
```

2. (5%) Show the weights and intercept of your model.



```
Weights: [-0.05401678 -1.18432309  0.99181238 -0.11744562  0.03172607 -0.65271009], Intercept: -0.09913337804277682
```

3. (10%) Show the accuracy score of your model on the testing set. The accuracy score should be greater than 0.75.



```
Accuracy: 0.7540983606557377
```

**(35%) Fisher's Linear Discriminant (FLD)**

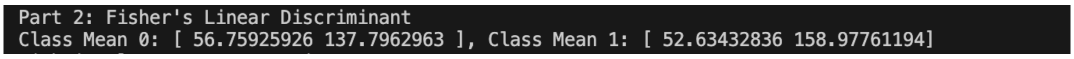
**Requirements:**

- Implement FLD to reduce the dimension of the data from 2-dimensional to 1-dimensional.

**Criteria:**

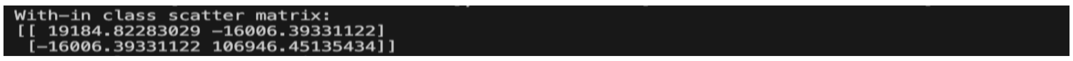
4. (0%) Show the mean vectors $m_i$ (i=0, 1) of each class of the training set.

(next page)



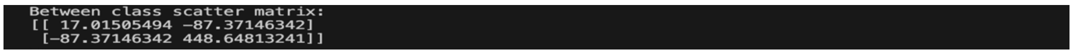
```
Part 2: Fisher's Linear Discriminant
Class Mean 0: [ 56.75925926 137.7962963 ], Class Mean 1: [ 52.63432836 158.97761194]
```

5. (5%) Show the within-class scatter matrix $S_W$ of the training set.



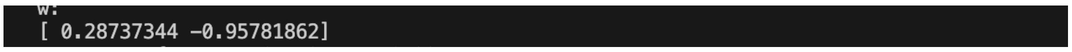
```
With-in class scatter matrix:
[[ 19184.82283029 -16006.39331122]
 [-16006.39331122 106946.45135434]]
```

6. (5%) Show the between-class scatter matrix $S_B$ of the training set.



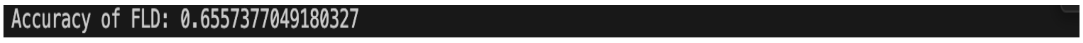
```
Between class scatter matrix:
[[ 17.01505494 -87.37146342]
 [-87.37146342 448.64813241]]
```

7. (5%) Show the Fisher's linear discriminant $w$ of the training set.



```
w:
[ 0.28737344 -0.95781862]
```

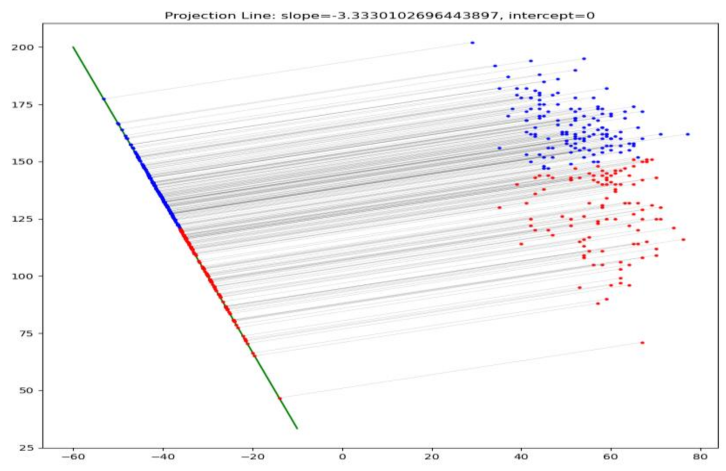
8. (10%) Obtain predictions for the testing set by measuring the distance between the projected value of the testing data and the projected means of the training data for the two classes. Show the accuracy score on the testing set. The accuracy score should be greater than 0.65.



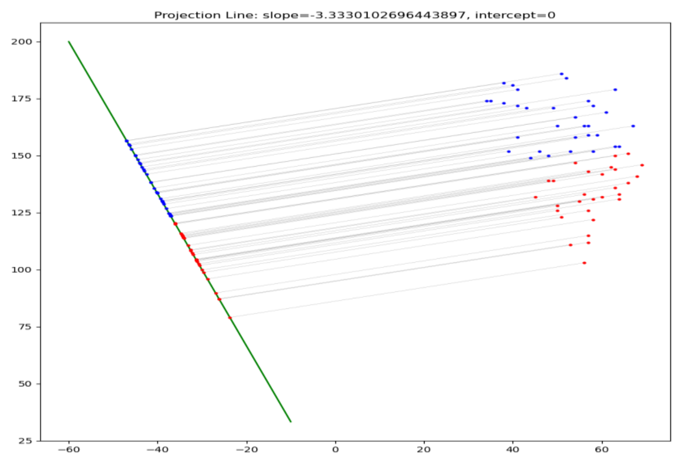
```
Accuracy of FLD: 0.6557377049180327
```

(next page)

9. (10%) Plot the projection line (x-axis: age, y-axis: thalach).
   **1)** Plot the projection line trained on the training set and show the slope and intercept on the title (you can choose any value of intercept for better visualization).

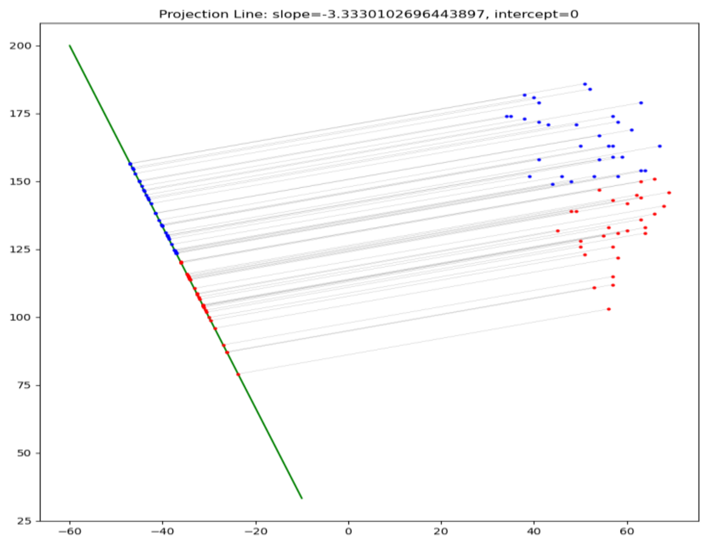


   **2)** Obtain the prediction of the testing set, plot and colorize them based on the prediction.



   **3)** Project all testing data points on your projection line. Your result should look like the below image.

(next page)



**Part. 2, Questions (50%):**

1. (5%) What's the difference between the sigmoid function and the softmax function? In what scenarios will the two functions be used? Please at least provide one difference for the first question and answer the second question respectively.

   **<u>Answer</u>**

   The sigmoid function is an activation function that maps any input value between 0 and 1. It is frequently used in the output layer of a binary classification problem, where the output is a probability that the input belongs to one of two classes. In such cases, the sigmoid function is a good option for the output layer because it generates a probability value that can be read as the likelihood of the input belonging to a particular class.

   The softmax function is an activation function that maps any input vector to a probability distribution. It is often used in the output layer of a multi-class classification problem, where the output should be a probability distribution over all classes. In such cases, the softmax function is a good option because it ensures that the output values are all positive and sum up to 1, which can be interpreted as a probability distribution.

   Both the sigmoid and softmax functions are commonly used in neural networks, but in different types of layers and for various purposes.

   In summary, the sigmoid function is used in the output layer of a binary classification problem to produce a probability value, while the softmax function is used in the output layer of a multi-class classification problem to produce a

(next page)

probability distribution over all the classes.

2. (10%) In this homework, we use the cross-entropy function as the loss function for Logistic Regression. Why can’t we use Mean Square Error (MSE) instead? Please explain in detail.

**Answer**

While Mean Square Error can potentially be used as a loss function in logistic regression, it is not the best choice for binary classification tasks. Cross-entropy (log loss) is more appropriate for the nature of the problem, gives a more interpretable measure of performance, is easier to optimize, and is more robust to imbalanced data. As a result, cross-entropy is the best choice for logistic regression.

(next page)

3. (15%) In a multi-class classification problem, assume you have already trained a classifier using a logistic regression model, which the outputs are P1, P2, ... Pc, how do you evaluate the overall performance of this classifier with respect to its ability to predict the correct class?

    3.1. (5%) What are the metrics that are commonly used to evaluate the performance of the classifier? Please at least list three of them.

    **Answer**

    Evaluating a classifier's performance is crucial to assessing how well it predicts. There are various commonly used metrics for this purpose, and I'll list three of them:

    Accuracy:

    Accuracy is a widely used metric that evaluates the proportion of correctly classified examples in relation to the total number of instances in the dataset. It gives a general overview of the classifier's performance. When dealing with imbalanced datasets, where one class greatly outnumbers the others, accuracy can be misleading.

    Precision and recall:

    Precision measures the proportion of true positive predictions (correctly predicted positive cases) out of all positive predictions produced by the classifier. Recall measures the proportion of true positive predictions out of all actual positive instances. These metrics are very effective in binary classification problems.

    F1 Score:

    The harmonic mean of precision and recall is the F1 score. It combines precision and recall into a single metric to provide a balance. The F1 score is especially useful when attempting to find a balance between making accurate positive predictions and capturing as many actual positives as possible.

    3.2. (5%) Based on the previous question, how do you determine the predicted class of each sample?

    **Answer**

    In a multi-class classification problem, the class with the maximum predicted probability (P1, P2..., Pc) is often chosen as the predicted class for that sample using logistic regression or similar models. In other words, the class with the maximum probability is predicted.

Mathematically, for a given sample, you find the class c for which Pc is the highest among all classes:

$$Predicted\_Class = \text{argmax}_i(P_i)$$

The class with the highest predicted probability is considered the predicted class for that particular sample.

3.3. (5%) In a class imbalance dataset (say 90% of class-1, 9% of class-2, and 1% of class-3), is there any problem with using the metrics you mentioned above and how to evaluate the model prediction performance in a fair manner?

**Answer**

Using traditional metrics such as accuracy, precision, recall, and F1 score in a multi-class classification problem with a highly imbalanced dataset can be difficult, especially for minority classes.

Accuracy:

Accuracy can be misleading, as it may be dominated by the majority class. In an imbalanced dataset, a high accuracy can be achieved by simply predicting the majority class for most samples.

Precision and Recall: Precision and recall for minority classes may be low due to the imbalance, which can be problematic in scenarios where correctly identifying rare classes is essential.

To evaluate the model's prediction performance in a fair manner in the presence of class imbalance, consider the following approaches:

Use Class-Specific Metrics:

Calculate precision, recall, and F1 score separately for each class to assess the model's performance for individual classes. This allows you to identify how well the classifier performs for minority classes.

Macro-Averaging and Micro-Averaging:

Compute macro-averaged and micro-averaged metrics. Macro-averaging computes the metric for each class and then takes the average, treating all classes equally. Micro-averaging aggregates the confusion matrix across all classes and then computes the metric. Micro-averaging gives more weight to the performance of the majority class.

For the training data, if we are training imbalanced data, we can use the following 3 methods :

Resampling Techniques:

Explore resampling techniques such as oversampling the minority classes, undersampling the majority class, or generating synthetic data to balance the class distribution.

Cost-sensitive Learning:
Modify the loss function or misclassification costs to be class-sensitive, assigning different costs to different classes based on their importance or rarity.

Ensemble Methods: Use ensemble methods, such as bagging or boosting, to combine multiple models and improve predictive performance on imbalanced datasets.

The choice of evaluation metrics and techniques should be guided by the specific goals and requirements of your multi-class classification problem, as well as the nature of the dataset and the importance of different classes.

4. (20%) Calculate the results of the partial derivatives for the following equations. (The first one is binary cross-entropy loss, and the second one is mean square error loss followed by a sigmoid function. σ is the sigmoid function.)

4.1. (10%)

$$\frac{\partial}{\partial x}\left(-t * \ln(\sigma(x)) - (1-t) * \ln(1-\sigma(x))\right)$$

$$\frac{\partial}{\partial x}\left(-t * \ln(\sigma(x)) - (1-t) * \ln(1-\sigma(x))\right) =$$

$$\frac{-t}{\sigma(x)} \times \sigma'(x) - \frac{(1-t)}{(1-\sigma(x))} \times \sigma'(x) \quad (1)$$

We need to compute the derivative of the sigmoid function, $\sigma(x)$, with respect to $x$

$$\sigma'(x) = \frac{d}{dx}\sigma(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right)$$

$$\sigma'(x) = \frac{1}{(1+e^{-x})} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) = \sigma(x) \cdot (1-\sigma(x)) \quad (2)$$

Replace $\sigma'(x)$ by its value in (1)

$$\frac{-t}{\sigma(x)} \cdot (\sigma(x)(1-\sigma(x))) - \frac{(1-t)}{(1-\sigma(x))} \cdot \sigma(x)(1-\sigma(x)) =$$

$$\underline{\underline{-t\,(1-\sigma(x)) - (1-t) \cdot \sigma(x)}}$$

4.2. (10%)

$$\frac{\partial}{\partial x}((t-\sigma(x))^2)$$

$$\frac{\partial}{\delta x}((t-\sigma(x))^2) =$$

$$-2(t-\sigma(x)) \cdot \sigma'(x) \qquad (1)$$

$$\sigma'(x) = \frac{d}{dx}\sigma(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \sigma(x)(1-\sigma(x)) \quad (2)$$

replace $\sigma'(x)$ by its value in (1)

$$\underline{\underline{-2(t-\sigma(x)) \cdot \sigma(x)(1-\sigma(x))}}$$