NYCU Introduction to Machine Learning, Homework 2

[111550151], [徐嘉亨]

Part. 1, Coding (60%):

(25%) Logistic Regression w/ Gradient Descent Method

1. (5%) Show the hyperparameters (learning rate and iteration, etc) that you used and the weights and intercept of your model.

```
LR = LogisticRegression(
    learning_rate=1e-3,
    num_iterations=1500,
)
```

```
2024-10-28 18:27:58.420 | INFO     | __main__:main:152 - LR: Weights: [-1.19664508  0.41205102  0.68097637 -1.00634739  0.24350775], Intercep: -3.166
6210890571684
```

2. (5%) Show the AUC of the classification results on the testing set.

```
AUC=0.8568
```

3. (15%) Show the accuracy score of your model on the testing set

```
LR: Accuracy=0.8333
```

(25%) Fisher Linear Discriminant, FLD

4. (5%) Show the mean vectors m_i (i=0, 1) of each class, the within-class scatter matrix *Sw,* and the between-class scatter matrix *Sb* of the training set.

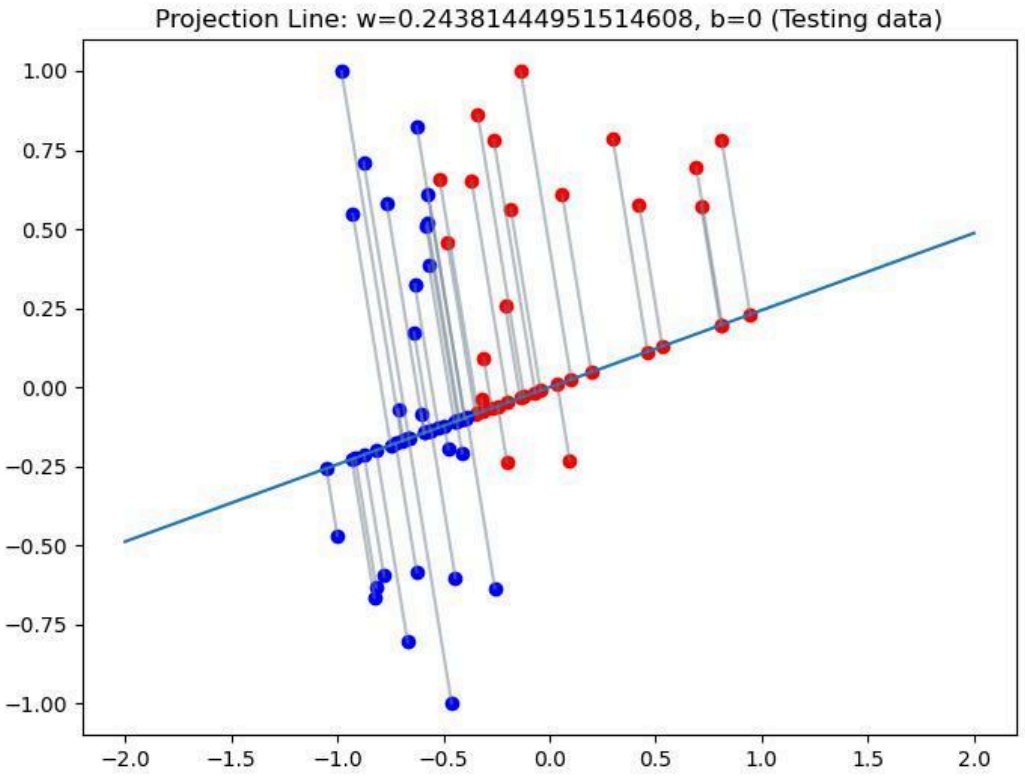
```
2024-10-28 18:27:58.427 | INFO     | __main__:main:175 - FLD: m0=[-0.27747695  0.29565197], m1=[-0.58535466  0.02331584] of cols=['10', '20']
2024-10-28 18:27:58.427 | INFO     | __main__:main:176 - FLD:
Sw=
[[17.17974856  5.44299487]
 [ 5.44299487 44.81848741]]
2024-10-28 18:27:58.427 | INFO     | __main__:main:177 - FLD:
Sb=
[[0.09478869 0.08384622]
 [0.08384622 0.07416696]]
```

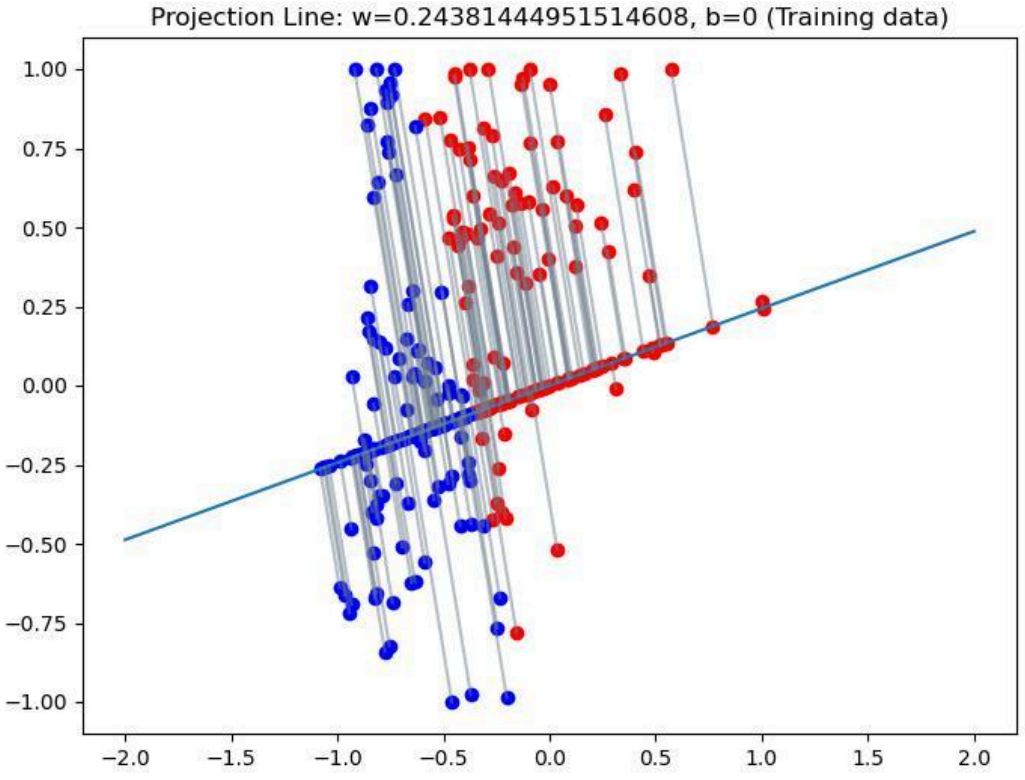
5. (5%) Show the Fisher's linear discriminant *w* of the training set.

```
2024-10-28 18:27:58.427 | INFO     | __main__:main:178 - FLD:
w=
[[-0.97154001]
 [-0.23687549]]
```

6. (15%) 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. (Also, plot for training data). Show the accuracy score on the testing set.

```
2024-10-28 18:27:58.427 | INFO     | __main__:main:179 - FLD: Accuracy=0.7619
```


Projection Line: w=0.24381444951514608, b=0 (Testing data)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Projection Line: w=0.24381444951514608, b=0 (Training data)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

(10%) Code Check and Verification

7. (10%) Lint the code and show the PyTest results.

```
(mlhw1) jxea666@LAPTOP-AR88PBDG:~/Intro2ML/hw2$ flake8 main.py
(mlhw1) jxea666@LAPTOP-AR88PBDG:~/Intro2ML/hw2$ pytest ./test_main.py -s
============================================================ test session starts ============================================================
platform linux -- Python 3.9.19, pytest-7.4.4, pluggy-1.0.0
rootdir: /home/jxea666/Intro2ML/hw2
collected 2 items

test_main.py (395, 2) (395,)
2024-10-28 18:36:38.463 | INFO     | test_main:test_logistic_regression:35 - accuracy=1.0000
.(395, 2) (395,)
2024-10-28 18:36:38.465 | INFO     | test_main:test_fld:45 - accuracy=0.8759
.

============================================================ 2 passed in 5.13s =============================================================
```

Part. 2, Questions (40%):

1. (10%)
   - Is logistic ‘regression’ used for regression problems?
   - If not, what task is primarily used? (without any additional techniques and modification); If yes, how can it be implemented?
   - Why are we using the logistic function in such a task? (list two reasons)
   - If there are multi-class, what should we use to substitute it?

   Although its name has ‘regression’, it is not used for regression problems. It is a classification technique and primarily used for binary classification tasks. The reasons why we are using the logistic function in a binary classification task are because the logistic function will transfer the input value to a real number in [0, 1], and we can see the result as a probability, which is ideal for binary classification. Another reason is that it introduces non-linearity to the model, enabling it to handle cases where classes are not linearly separable. If there are multi-class cases, we should use softmax function to substitute the logistic function to handle the multi-class cases.

2. (15%) When a trained classification model shows exceptionally high precision but unusually low recall and F1-score, what potential issues might arise? How can these issues be resolved? List at least three solutions.

   First, the model may be trained on a dataset where one class is significantly underrepresented, causing it to be less sensitive to that class. Second, a high decision threshold (e.g., 0.8 instead of 0.5) may result in very few positive predictions, increasing precision but lowering recall. Third, if the model is overfitting to the training data, it may be less generalizable and may fail to capture a sufficient range of positive instances. Solutions: 1. Lower the decision threshold for positive classification. For example, if the current threshold is 0.8, lowering it to 0.5 can increase recall by allowing more positives to be classified as such. 2. Apply regularization to reduce overfitting, making the model more robust in capturing positives. 3. Use cross-validation to test the model on multiple splits of the data, which helps ensure it performs consistently and generalizes well. 4. Prevent imbalanced training dataset. 5. Assign higher weights to the minority class, which helps the model give more attention to underrepresented instances.

3. (15%) In this homework, we use Cross-Entropy as the loss function for Logistic Regression. Can we use Mean Square Error (MSE) instead? Why or why not? Please explain in detail.

Although MSE can be used for logistic regression, it is not recommended. MSE is more suitable for regression problems than classification problems. The reason why we should use CE instead of MSE for logistic regression is because in logistic regression, the model output is a probability in [0, 1] obtained by applying the sigmoid function to a linear combination of inputs. When using MSE, the gradients don't align well with the goal of minimizing classification error, as MSE is optimized for continuous values rather than probabilities. This mismatch leads to less efficient learning, with updates that don't minimize the probability-based error effectively. And CE loss increases as the predicted probability diverges from the true label, causing more substantial weight updates for incorrect predictions. In contrast, MSE provides smaller updates even when the predictions are far from the true label, leading to slower convergence. Finally, CE directly measures the distance between the predicted probabilities and the actual binary labels, making it a more interpretable measure of classification accuracy. MSE, on the other hand, lacks a probabilistic interpretation, as it treats the output more like a regression prediction than a probability. In short, using MSE for logistic regression can lead to slower, less effective learning, and make interpreting the loss in a classification context more difficult.