NYCU Introduction to Machine Learning, Homework 2

[111550088], [張育維]

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-2,  # You can modify the parameters as you want
    num_iterations=10000,  # You can modify the parameters as you want
)
```

```
2024-10-29 11:09:59.191 | INFO     | __main__:main:183 - LR: Weights: [[-0.50076954]
 [ 0.10884352]
 [ 0.60821965]
 [ 0.06066269]
 [ 0.13523584]], Intercep: -1.814800433097112
```

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

```
2024-10-27 23:44:05.631 | INFO     | __main__:main:184 - LR: Accuracy=0.8095, AUC=0.8500
```

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

```
2024-10-27 23:44:05.631 | INFO     | __main__:main:184 - LR: Accuracy=0.8095, AUC=0.8500
```

(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.

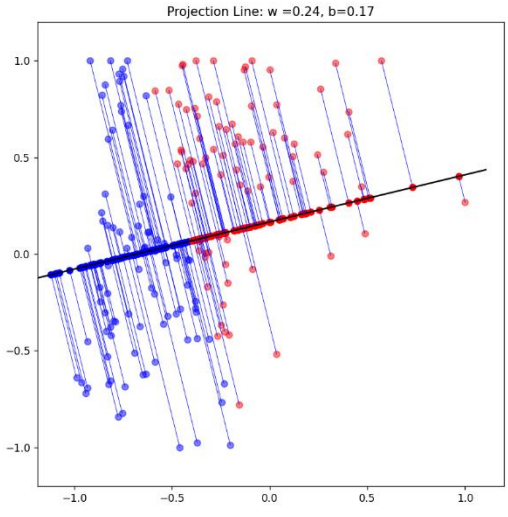
```
2024-10-27 23:44:05.634 | INFO     | __main__:main:206 - FLD: m0=[[-0.27747695  0.29565197]], m1=[[-0.58535466  0.02331584]] of
 cols=['10', '20']
2024-10-27 23:44:05.634 | INFO     | __main__:main:207 - FLD:
Sw=
[[17.17974856  5.44299487]
 [ 5.44299487 44.81848741]]
2024-10-27 23:44:05.634 | INFO     | __main__:main:208 - FLD:
Sb=
[[0.09478869 0.08384622]
 [0.08384622 0.07416696]]
```

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

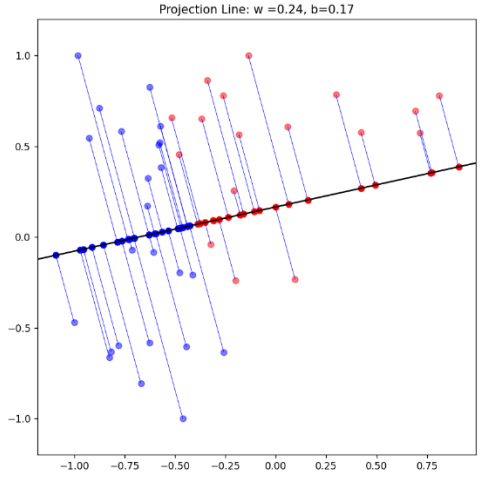
```
2024-10-27 23:44:05.635 | INFO     | __main__:main:209 - FLD:
W=
[[-0.0166359 ]
 [-0.00405607]]
```

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.

```
2024-10-27 23:44:05.635 | INFO     | __main__:main:210 - FLD: Accuracy=0.7619
```



The graph of train data



The graph of test data

(10%) Code Check and Verification

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

```
(machine_learning) D:\user\Desktop\for_school\machine_learning\hw2>flake8 main.py

(machine_learning) D:\user\Desktop\for_school\machine_learning\hw2>
```

```
(machine_learning) D:\user\Desktop\for_school\machine_learning\hw2>pytest ./test_main.py -s
============================================= test session starts =============================================
platform win32 -- Python 3.11.9, pytest-8.3.3, pluggy-1.5.0
rootdir: D:\user\Desktop\for_school\machine_learning\hw2
collected 2 items

test_main.py (395, 2) (395,)
2024-10-27 23:46:40.595 | INFO     | test_main:test_logistic_regression:35 - accuracy=0.9517
.(395, 2) (395,)
2024-10-27 23:46:40.600 | INFO     | test_main:test_fld:45 - accuracy=0.8759
.

============================================== 2 passed in 14.60s ==============================================
```

Part. 2, Questions (40%):

1. (10%)

- Is logistic ‘regression’ used for regression problems?
- If not, what task is it 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?

No, logistic regression is used for classification, not regression. In logistic regression, the model predicts a categorical outcome, typically binary (0 or 1), by estimating the probability that a given input belongs to a particular class. The term 'regression' in the name refers to the linear relationship applied before using the logistic (sigmoid) function. However, the primary purpose of logistic regression is classification.

It is mainly used for classification, predicting the probability that a given input belongs to a certain class by mapping the output to a value between 0 and 1.

1. Probability Output: Using the logistic function, we can squash all real-valued inputs into a range between 0 and 1, making the output easily interpretable as a probability. This allows us to assign confidence to predictions.
2. Non-linearity and Decision Boundary: The logistic function introduces non-linearity, enabling logistic regression to model binary outcomes effectively. It creates a smooth decision boundary that allows for clear and adjustable classification thresholds, providing more confident classifications.

We can extend logistic regression to handle multiple classes using softmax regression, which generalizes the sigmoid function to output a probability distribution over all classes. Alternatively, we can use the One-vs-Rest (OvR) approach, which creates a separate binary classifier for each class. Each classifier distinguishes one class from all others, and the model assigns the class with the highest probability output from any of the classifiers.

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.

This outcome makes the model overly conservative in making positive predictions. It can also lead to class imbalance, where the model becomes biased towards the majority class, resulting in poor predictive performance.

To address this issue, several solutions are available. First, we can lower the decision threshold to classify more cases as positive, which can increase recall and improve the F1-score. Next, resampling or class-weighting techniques can help the model better recognize minority cases. Additionally, feature engineering can be used to add relevant features, or dimensionality reduction can emphasize key variables. These approaches can improve recall without sacrificing too much precision.

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.

While MSE can technically be used as the loss function for logistic regression, it's not recommended. Logistic regression outputs probabilities for binary classification tasks, ranging from 0 to 1, whereas MSE is typically used for regression tasks with continuous outputs. Additionally, MSE is too sensitive to prediction errors due to squaring the differences, which can slow convergence during training and decrease learning efficiency. Therefore, although MSE can be applied mathematically, it is less effective than cross-entropy, which leads to a more efficient learning model.