NYCU Pattern Recognition, Homework 2

109550198, 卜銳凱

Part. 1, Coding (60%):

(25%) Logistic Regression w/ Gradient Descent Method (slide ref)

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

```
LR = LogisticRegression(
    learning_rate=0.00001,  # You can modify the parameters as you want
    num_iterations=3000000,  # You can modify the parameters as you want
)
```

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

```
2024-04-17 19:32:20.135 | INFO     | __main__:main:155 - LR: Weights: [-0.22138051  0.07617309  0.27272012  0.13996364  0.14498164], Intercep: -0.8893906755311495
```

3. (5%) Show the AUC of the classification results on the testing set.

```
2024-04-17 19:32:20.143 | INFO     | __main__:main:156 - LR: Accuracy=0.8095, AUC=0.8477
```

4. (15%) Show the accuracy score of your model on the testing set.

```
2024-04-17 19:32:20.143 | INFO     | __main__:main:156 - LR: Accuracy=0.8095, AUC=0.8477
```

(25%) Fisher Linear Discriminant, FLD (slide ref)

5. (0%) Show the mean vectors mi (i=0, 1) of each class of the training set.

```
2024-04-17 19:32:20.170 | INFO     | __main__:main:169 - FLD: m0=[ 0.35994138 -0.04560139], m1=[0.32519126 0.04435118]
```

6. (5%) Show the within-class scatter matrix *Sw* and between-class scatter matrix *Sb* of the training set.

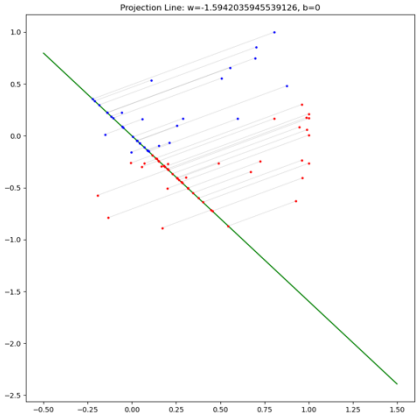
```
Sw=
[[41.93041055 15.7202037 ]
 [15.7202037  37.25186904]]
2024-04-17 19:32:20.171 | INFO     | __main__:main:171 - FLD:
Sb=
[[ 0.00120757 -0.00312586]
 [-0.00312586  0.00809147]]
2024-04-17 19:32:20.171 | INFO     | __main__:main:172 - FLD:
```

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

```
w=
[ 0.53138255 -0.84713198]
2024-04-17 19:32:20.171 | INFO     | __main__:main:173 - FLD: Accuracy=0.7381
```

8. (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. Show the accuracy score on the testing set.

```
__main__:main:173 - FLD: Accuracy=0.7381
```



(10%) Code Check and Verification

9. (10%) Lint the code and show the PyTest results.

```
============================= test session starts ==============================
platform darwin -- Python 3.9.13, pytest-7.1.2, pluggy-1.0.0
rootdir: /Users/ralphkedywillensbuteau/Desktop/PR/HW2
plugins: anyio-3.5.0
collected 2 items

test_main.py (395, 2) (395,)
2024-04-17 19:48:41.999 | INFO     | test_main:test_logistic_regression:35 - accuracy=0.9517
.(395, 2) (395,)
2024-04-17 19:48:42.002 | INFO     | test_main:test_fld:45 - accuracy=0.8759
.

============================== 2 passed in 6.96s ===============================
```

Part. 2, Questions (40%):

1. (10%) Is it suitable to use Mean Square Error (MSE) as the loss function for Logistic Regression? Please explain in detail.

   Write or type your answer here.

   Using Mean Squared Error (MSE) as a loss function for logistic regression is generally unsuitable due to several key issues:
   - Mismatch with Probabilistic Outputs: Logistic regression predicts probabilities, and MSE does not fit this because it does not handle probabilities effectively.
   - Non-Convexity: In the context of logistic regression, MSE generates a non-convex optimization problem, which may make it difficult to locate the global minimum due to several local minima.
   - Gradient Vanishing: When MSE is combined with logistic regression, the gradients can become quite tiny during training, especially when predictions are close to the class boundaries (0 or 1). This can slow or stop convergence.
   - Inappropriate Penalties: MSE does not penalize classification errors in a way that emphasizes the probabilistic nature of the outputs, unlike binary cross-entropy, which provides more meaningful feedback for classification tasks.

   Therefore, binary cross-entropy is the preferred loss function for logistic regression as it better suits the model's probabilistic nature and the objectives of classification.

2. (15%) In page 31 of the lecture material (linear_classification.pdf), we introduce two methods for performing classification tasks using Fisher's linear discriminator: 1) Determining a threshold, 2) Using the k-NN (k-nearest neighbors) rule. Please discuss at least three aspects, either advantages or disadvantages, of using the k-NN method compared to determining a threshold (resources, performance, etc.).

Write or type your answer here.

When comparing the k-nearest neighbors (k-NN) method to determining a threshold for classification with Fisher's linear discriminant, three key aspects to consider are:

Computational Complexity and Resource Usage:

k-NN: Computationally intensive and requires storing the entire dataset, leading to high memory use and slower predictions. However, it easily adapts to new data since there's no explicit training phase.
Threshold Method: Once trained, making predictions is computationally cheap and fast, as it involves a simple calculation and comparison.
Flexibility and Adaptability:

k-NN: Highly adaptable and flexible, handling complex decision boundaries well since it makes no assumptions about data distribution.
Threshold Method: Less flexible, assuming normally distributed data within each class, which can limit performance if these conditions are not met.
Performance in Different Scenarios:

k-NN: Can struggle with high-dimensional data (curse of dimensionality) and requires careful tuning of parameters, but excels in handling irregular decision boundaries in small to medium datasets.
Threshold Method: Performs best when classes are linearly separable or nearly so, benefiting from simplicity and robustness when data distributions align with its assumptions.
These distinctions highlight the importance of considering dataset characteristics and specific task requirements when choosing between k-NN and a threshold-based approach in Fisher's linear discriminant analysi

3. (15%) In logistic regression, what is the relationship between the sigmoid function and the softmax function? In what scenarios will the two functions be used respectively?

Write or type your answer here.

The relationship between the sigmoid and softmax functions in logistic regression centers on their roles in handling classification probabilities:

The sigmoid function is used in binary classification within logistic regression to map any real number to a probability between 0 and 1, hence predicting the probability of one of two classes.

The Softmax function extends the sigmoid function to multi-class classification scenarios. It transforms a vector of values into a probability distribution over many classes, with each class probability proportional to the exponential of its score.

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 probability distribution over all the classes.