NYCU Pattern Recognition, Homework 2

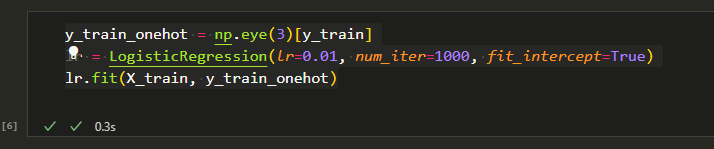
**411551005 徐浩哲**

**Part. 1, Coding (70%)**:

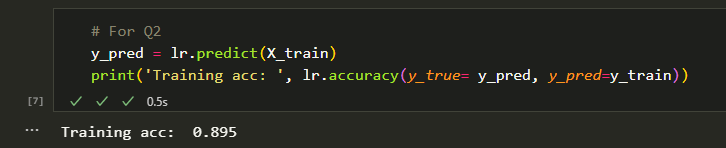
**(20%) Logistic Regression Model**

**Criteria:**

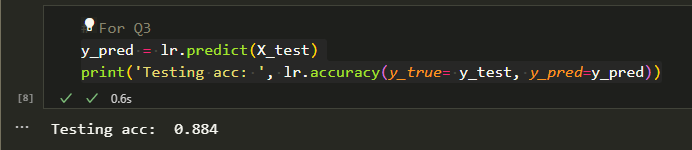
1. (0%) Show the learning rate, epoch, and batch size that you used.



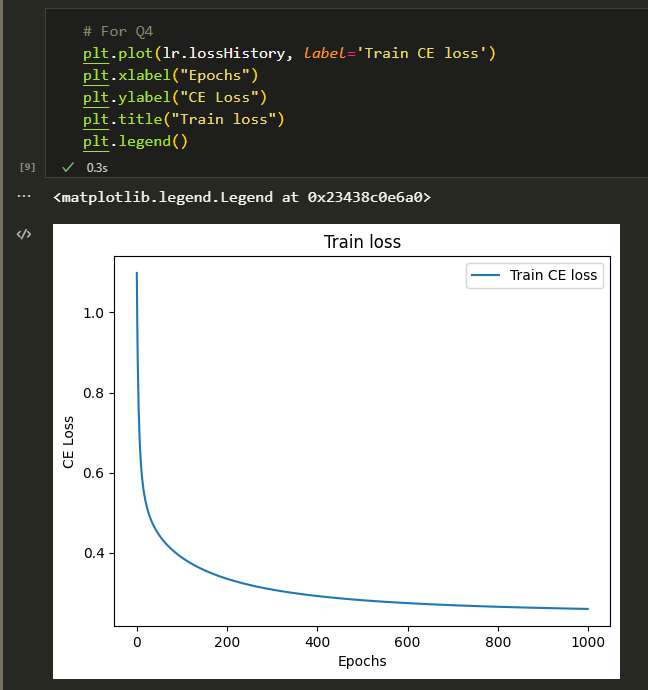
1. (5%) What’s your training accuracy?



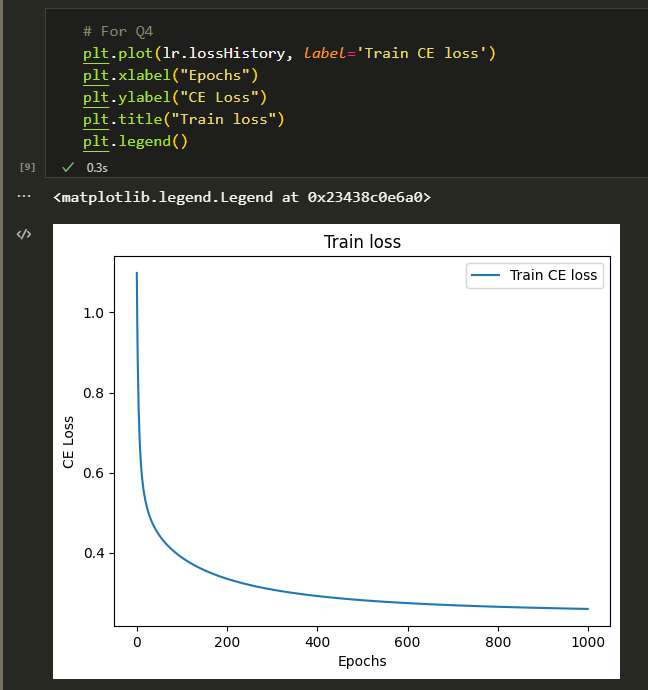
1. (5%) What’s your testing accuracy?



1. (5%) Plot the learning curve of the training. (x-axis=epoch, y-axis=loss)



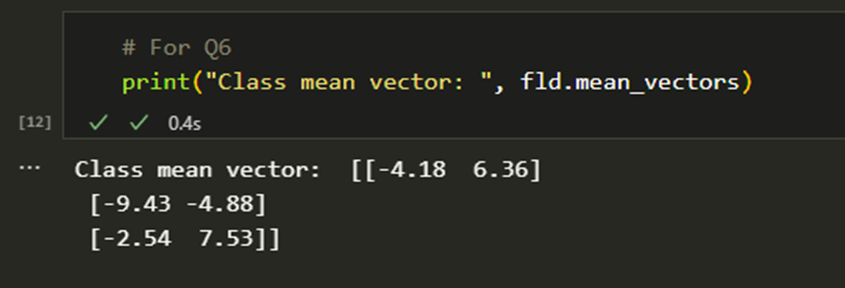
1. (5%) Show the [confusion matrix](https://ycc.idv.tw/confusion-matrix.html) on testing data.



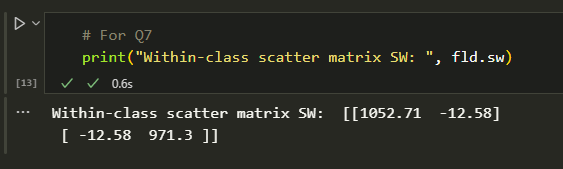
**(30%) Fisher’s Linear Discriminant (FLD) Model**

**Criteria:**

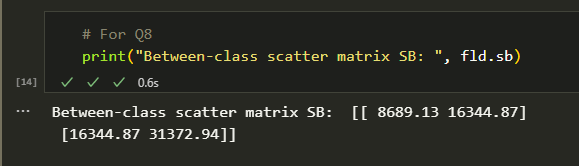
1. (2%) Compute the mean vectors (i=1, 2, 3) of each class on training data.



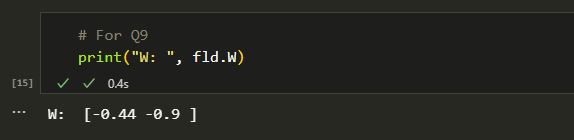
1. (2%) Compute the within-class scatter matrix on training data.



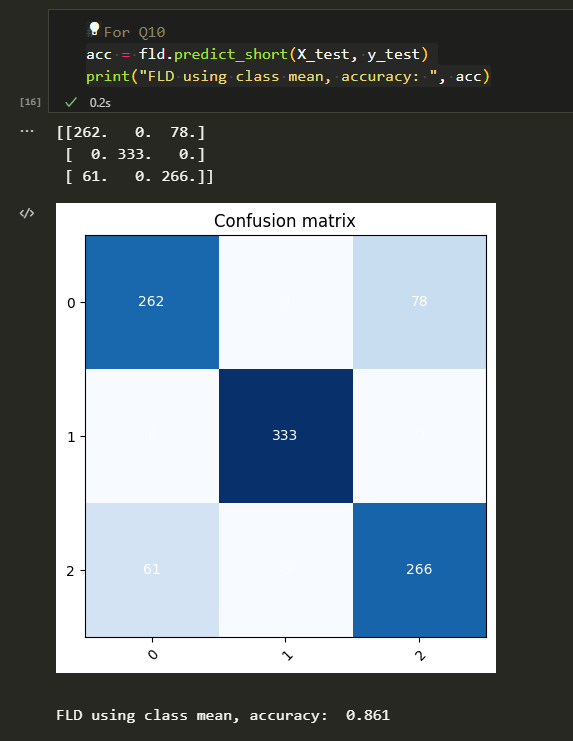
1. (2%) Compute the between-class scatter matrix on training data.



1. (4%) Compute the Fisher’s linear discriminant on training data.



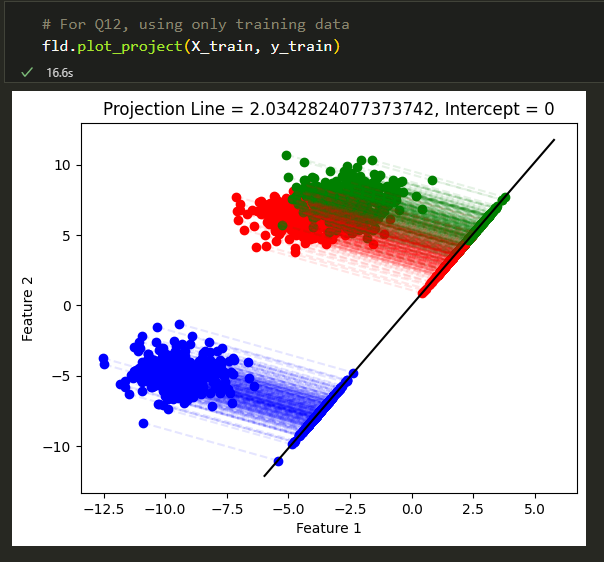
1. (8%) Project the testing data to get the prediction using the shortest distance to the class mean. Report the accuracy score and draw the confusion matrix on testing data.



1. (8%) Project the testing data to get the prediction using [K-Nearest-Neighbor](https://www.ibm.com/topics/knn). Compare the accuracy score on the testing data with K values from 1 to 5.

|  |  |  |
| --- | --- | --- |
| k = 1 | k = 2 | k = 3 |
|  |  |  |
| k = 4 | k = 5 |  |
|  |  |  |

1. (4%) **1)** Plot the best projection line on the training data and show the slope and intercept on the title *(you can choose any value of intercept for better visualization)***2)** colorize the training data with each class **3)** project all training data points on your projection line. Your result should look like the below image (This image is for reference, not the answer)



**(20%) Train your own model**

**Requirements:**

**Criteria:**

1. Explain how you chose your model and what feature processing you have done in detail. Otherwise, no points will be given.

|  |  |
| --- | --- |
| Point | Accuracy |
| 20 | testing acc > 0.921 |
| 15 | 0.91 <= testing acc <= 0.921 |
| 8 | 0.9 <= testing acc <= 0.91 |
| 0 | testing acc < 0.9 |

**Part. 2, Questions (30%):**

(6%) 1. Discuss and analyze the performance a) between Q10 and Q11, which approach is more suitable for this dataset. Why? b) between different values of k in Q11. (Which is better, a larger or smaller k? Does this always hold?)

1. The suitability of using the shortest distance to the class mean or KNN approach depends on the specific characteristics of the dataset and the problem at hand. If the dataset has well-defined clusters and the distance between the clusters is sufficiently large, the shortest distance to the class mean approach can be effective. This approach calculates the distance between each sample and the mean of its corresponding class, and assigns the sample to the class with the closest mean. This can be particularly useful when the decision boundary between the classes is relatively smooth. On the other hand, if the dataset has overlapping or non-linear decision boundaries, the KNN approach may be more suitable. KNN calculates the distance between a given sample and its k nearest neighbors in the training set, and assigns the sample to the class that is most common among its k nearest neighbors. This can be effective in capturing the local structure of the data and can perform well in cases where the decision boundary is complex. In conclusion, the choice between the shortest distance to the class mean and KNN approach depends on the specific characteristics of the dataset and the problem at hand. It is important to experiment with different approaches and evaluate their performance to determine the most suitable approach for a given problem.
2. The choice of the value of k in K-NN algorithm is a critical hyperparameter that can affect the performance of the model. The optimal value of k depends on the specific characteristics of the data and the problem at hand. In general, a larger value of k will result in a smoother decision boundary and may reduce the effects of noise in the data. This can lead to better performance on the test data and can help to prevent overfitting. On the other hand, a smaller value of k can capture more local details and may perform better on the training data. However, this is not always the case, as the optimal value of k can vary depending on the specific dataset and problem. In some cases, a small value of k may be optimal, while in others, a larger value of k may be preferred. Therefore, it is important to perform experiments with different values of k and evaluate the performance on a validation set to determine the optimal value for a given problem.

(6%) 2. Compare the sigmoid function and softmax function.

Both sigmoid and softmax functions are commonly used in machine learning, particularly in neural networks, to transform input values into a range of output probabilities. The sigmoid function is a mathematical function that maps any input value to a value between 0 and 1. Its formula is:

sigmoid(x) = 1 / (1 + e^(-x))

The sigmoid function is commonly used as an activation function in neural networks for binary classification problems, where the output is either 0 or 1. It has the property that its output is always between 0 and 1, which makes it useful for modeling probabilities.The softmax function is also a mathematical function that maps input values to a range of output probabilities. However, it is used for multi-class classification problems, where the output can be one of several possible classes. Its formula is:

softmax(x\_i) = e^(x\_i) / sum(e^(x\_j))

where x\_i is the input value for the i-th class, and the sum is taken over all classes.

The softmax function has the property that its output probabilities always sum to 1, which makes it useful for modeling the probability distribution over multiple classes. In summary, sigmoid function is used for binary classification problems, while softmax function is used for multi-class classification problems. Both functions transform input values into a range of output probabilities, but they differ in their formulas and properties.

(6%) 3. Why do we use cross entropy for classification tasks and mean square error for

regression tasks?

Cross-entropy and mean square error are both loss functions used in machine learning to measure the difference between predicted and actual values, but they are typically used for different types of tasks.

Cross-entropy is commonly used as a loss function for classification tasks because it measures the difference between predicted class probabilities and true class probabilities. The cross-entropy loss function is designed to penalize models that make incorrect predictions with high confidence. In other words, it encourages the model to assign high probabilities to the correct classes and low probabilities to the incorrect classes.

Mean square error, on the other hand, is commonly used as a loss function for regression tasks because it measures the difference between predicted and actual numerical values. The mean square error loss function is designed to penalize models that make large errors in their predictions. In other words, it encourages the model to minimize the difference between predicted and actual values.

The choice of loss function depends on the specific task and the type of output that the model is predicting. Classification tasks involve predicting a discrete label or category, while regression tasks involve predicting a continuous numerical value. Cross-entropy loss is more suitable for classification tasks, as it measures the difference between predicted probabilities and true probabilities, while mean square error is more suitable for regression tasks, as it measures the difference between predicted and actual numerical values.

(6%) 4. In Q13, we provide an imbalanced dataset. Are there any methods to improve Fisher Linear Discriminant's performance in handling such datasets?

(6%) 5. 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.)

