NYCU Pattern Recognition, Homework 2

Deadline: Apr. 08, 23:59

Part. 1, Coding (70%):

In this coding assignment, you are requested to implement 1) logistic Regression and 2) Fisher's Linear Discriminant by using only Numpy, then train your model on the provided dataset and finally evaluate the performance on testing data. Please train your logistic regression model using Gradient Descent, not the closed-form solution.

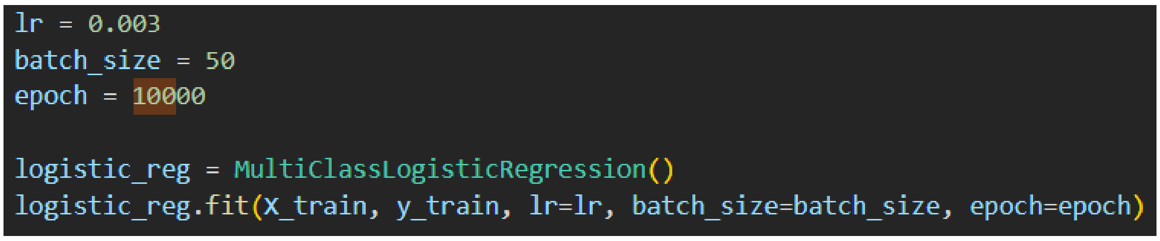
(20%) Logistic Regression Model

Requirements:

- Use Gradient Descent
- Use CE (Cross-Entropy) as your loss function.
- Use Softmax for this multiclass classification task.

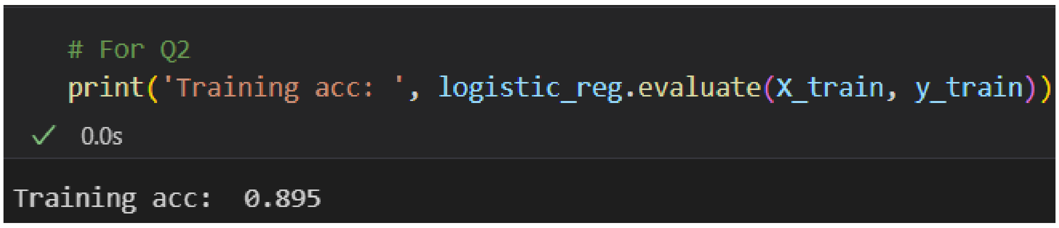
Criteria:

1. (0%) Show the learning rate, epoch, and batch size that you used.
   learning rate = 0.003
   batch size = 50
   epoch = 10000



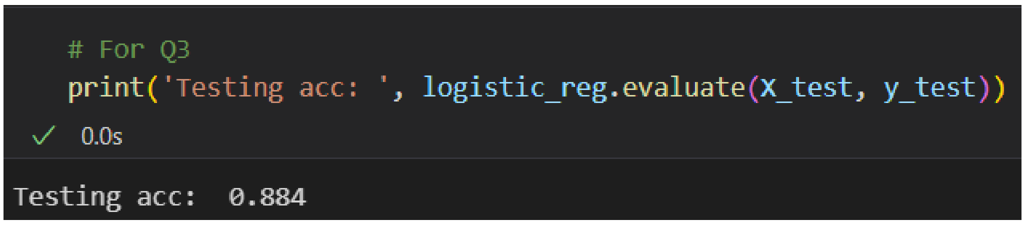
```
lr = 0.003
batch_size = 50
epoch = 10000

logistic_reg = MultiClassLogisticRegression()
logistic_reg.fit(X_train, y_train, lr=lr, batch_size=batch_size, epoch=epoch)
```

2. (5%) What's your training accuracy?
   Train accuracy = 0.895



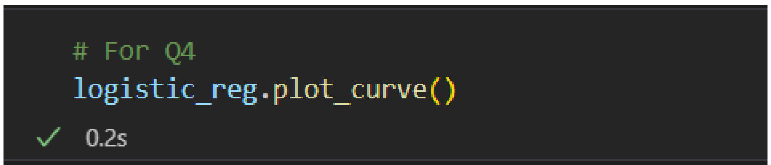
```
# For Q2
print('Training acc: ', logistic_reg.evaluate(X_train, y_train))
✓ 0.0s

Training acc:  0.895
```

3. (5%) What's your testing accuracy?
   Test accuracy = 0.884

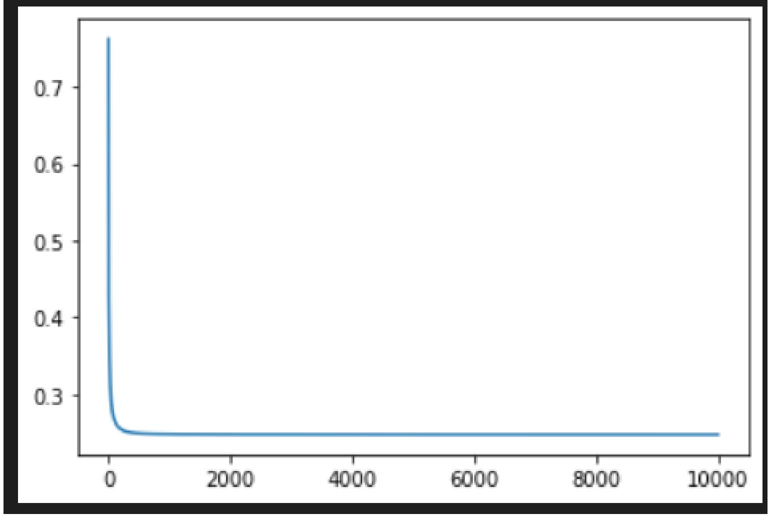


```
# For Q3
print('Testing acc: ', logistic_reg.evaluate(X_test, y_test))
✓ 0.0s

Testing acc:  0.884
```

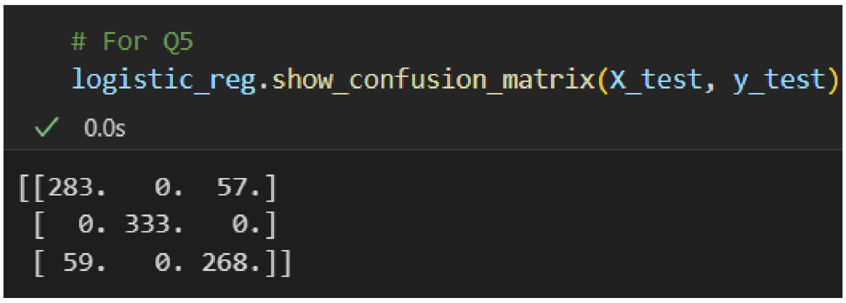
4. (5%) Plot the learning curve of the training. (x-axis=epoch, y-axis=loss)



```
# For Q4
logistic_reg.plot_curve()
```
✓ 0.2s



5. (5%) Show the confusion matrix on testing data.



```
# For Q5
logistic_reg.show_confusion_matrix(X_test, y_test)
```
✓ 0.0s

```
[[283.   0.  57.]
 [  0. 333.   0.]
 [ 59.   0. 268.]]
```

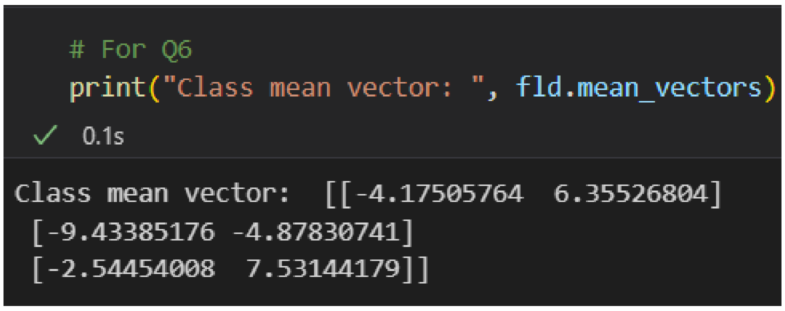
(30%) Fisher's Linear Discriminant (FLD) Model

Requirements:

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

Criteria:

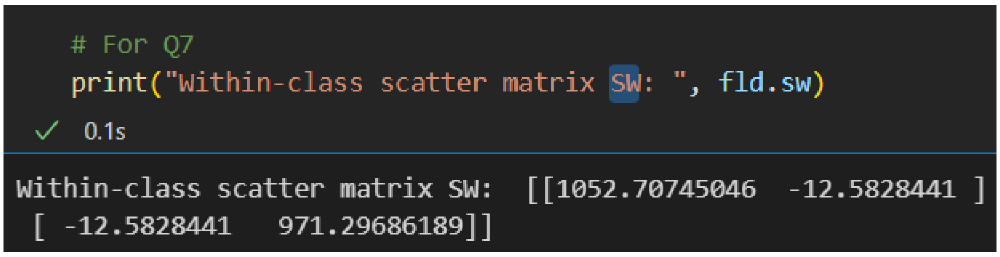
6. (2%) Compute the mean vectors M_i (i=1, 2, 3) of each class on training data.



```
# For Q6
print("Class mean vector: ", fld.mean_vectors)
```
✓ 0.1s

```
Class mean vector:  [[-4.17505764  6.35526804]
 [-9.43385176 -4.87830741]
 [-2.54454008  7.53144179]]
```

7. (2%) Compute the within-class scatter matrix S_W on training data.

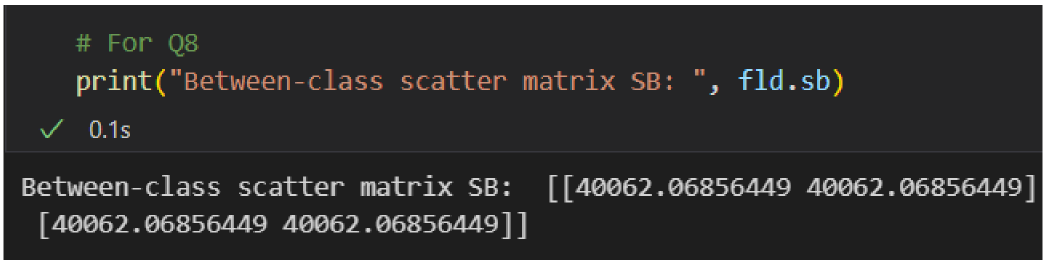


```
# For Q7
print("Within-class scatter matrix SW: ", fld.sw)
```

✓ 0.1s

```
Within-class scatter matrix SW:  [[1052.70745046  -12.5828441 ]
 [ -12.5828441   971.29686189]]
```

8. (2%) Compute the between-class scatter matrix S_B on training data.

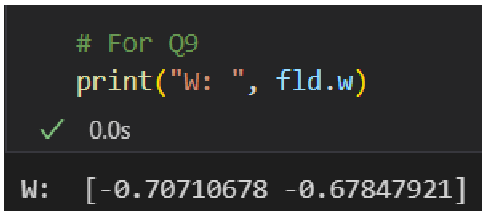


```
# For Q8
print("Between-class scatter matrix SB: ", fld.sb)
```

✓ 0.1s

```
Between-class scatter matrix SB:  [[40062.06856449 40062.06856449]
 [40062.06856449 40062.06856449]]
```

9. (4%) Compute the Fisher's linear discriminant W on training data.

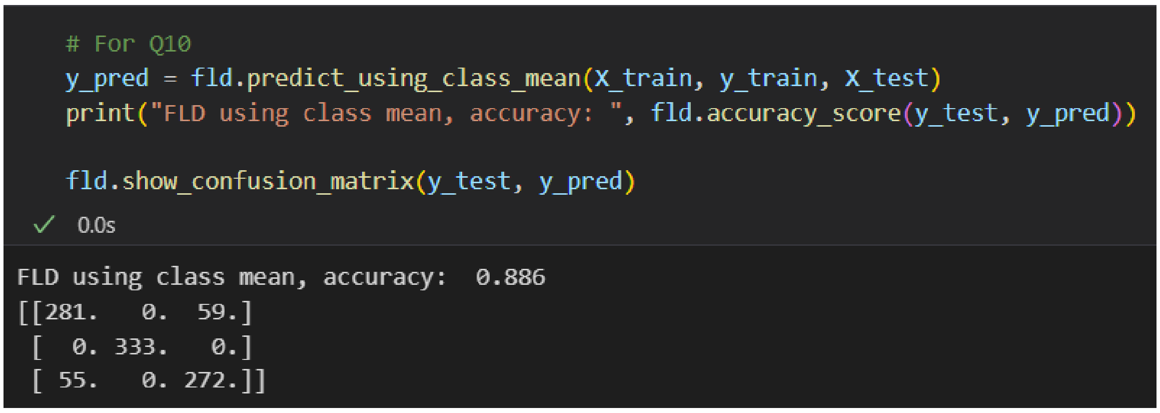


```
# For Q9
print("W: ", fld.w)
```

✓ 0.0s

```
W:  [-0.70710678 -0.67847921]
```

10. (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.

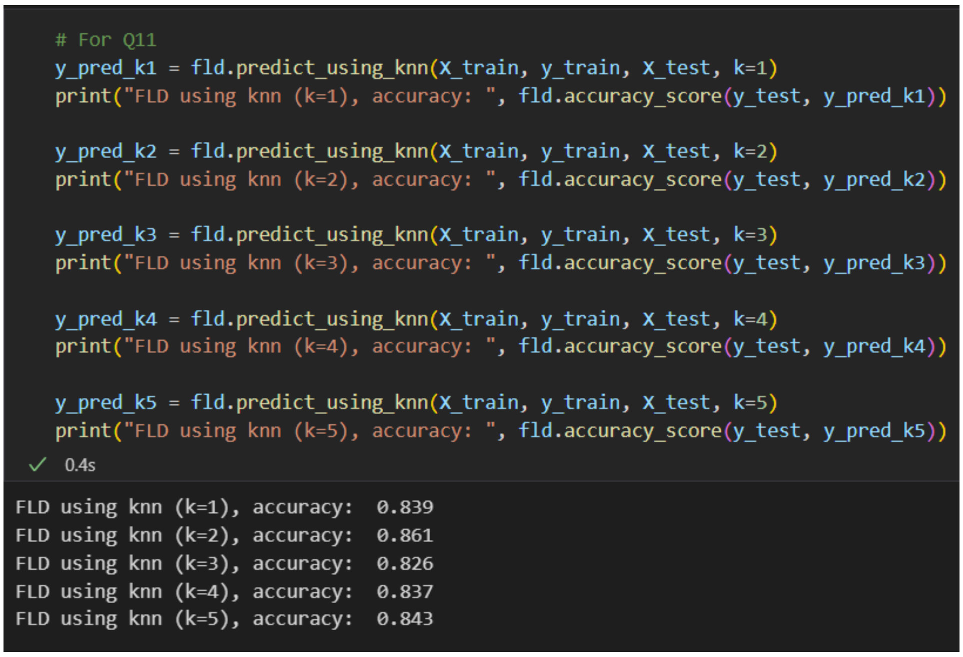


```
# For Q10
y_pred = fld.predict_using_class_mean(X_train, y_train, X_test)
print("FLD using class mean, accuracy: ", fld.accuracy_score(y_test, y_pred))

fld.show_confusion_matrix(y_test, y_pred)
```

✓ 0.0s

```
FLD using class mean, accuracy:  0.886
[[281.   0.  59.]
 [  0. 333.   0.]
 [ 55.   0. 272.]]
```

11. (8%) Project the testing data to get the prediction using K-Nearest-Neighbor. Compare the accuracy score on the testing data with K values from 1 to 5.

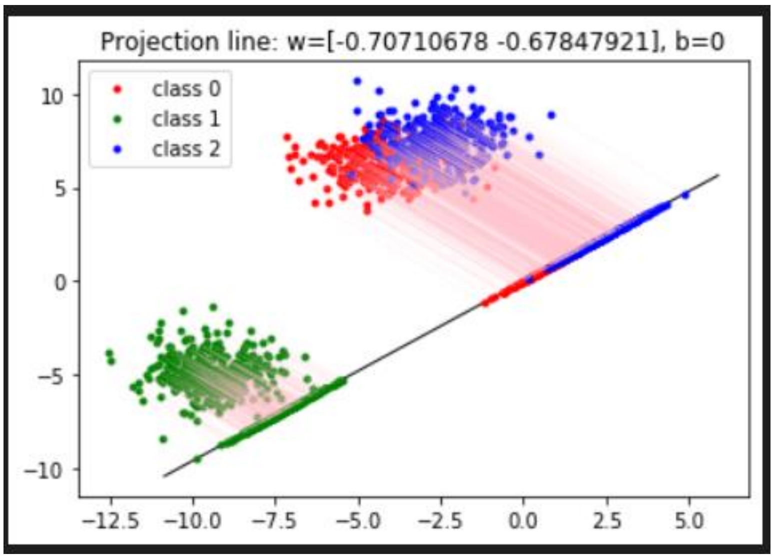


```
# For Q11
y_pred_k1 = fld.predict_using_knn(X_train, y_train, X_test, k=1)
print("FLD using knn (k=1), accuracy: ", fld.accuracy_score(y_test, y_pred_k1))

y_pred_k2 = fld.predict_using_knn(X_train, y_train, X_test, k=2)
print("FLD using knn (k=2), accuracy: ", fld.accuracy_score(y_test, y_pred_k2))

y_pred_k3 = fld.predict_using_knn(X_train, y_train, X_test, k=3)
print("FLD using knn (k=3), accuracy: ", fld.accuracy_score(y_test, y_pred_k3))

y_pred_k4 = fld.predict_using_knn(X_train, y_train, X_test, k=4)
print("FLD using knn (k=4), accuracy: ", fld.accuracy_score(y_test, y_pred_k4))

y_pred_k5 = fld.predict_using_knn(X_train, y_train, X_test, k=5)
print("FLD using knn (k=5), accuracy: ", fld.accuracy_score(y_test, y_pred_k5))
```

✓ 0.4s

```
FLD using knn (k=1), accuracy:  0.839
FLD using knn (k=2), accuracy:  0.861
FLD using knn (k=3), accuracy:  0.826
FLD using knn (k=4), accuracy:  0.837
FLD using knn (k=5), accuracy:  0.843
```

12. (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:

- Using another dataset that we provided (a real-world dataset).
- Train your model (FLD or Logistics Regression model that you implemented above).
- Try different parameters and feature engineering to beat the baseline.
- Save your testing predictions in the CSV file.

Criteria:

13. 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

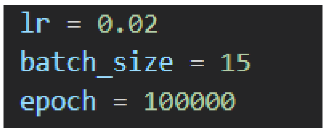
Model:

I choose Logistic regression as my model because I can use it without change anything in the model used for Q1 ~ Q5. Another reason is that I cannot image how the FLD activate with more than 2 dimensions.

Data preprocessing:

Multiply feature 1 and feature 2 by 100, feature 3 and feature 4 by 10. It is used to enlarge the distance between data to achieve the purpose of separating classes.

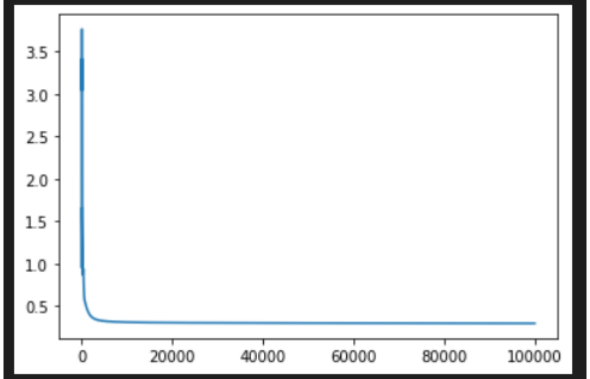
Hyper parameters:



```
lr = 0.02
batch_size = 15
epoch = 100000
```

Result:

```
Valuation acc:  0.910958904109589
```



```
[[324.   0.   5.]
 [  3. 144.   3.]
 [ 48.   6. 197.]]
```

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?

Ans:

FLD is more suitable because there are many data overlapping between class 0 and class 2. The label of data on border will be influenced by the data nearby them when using knn, so FLD is more suitable for this dataset.

b) between different values of k in Q11. (Which is better, a larger or smaller k? Does this always hold?)

Ans:

k=2 is better, k will be influenced by distribution of the data

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

Ans:

The sum of probabilities predicted by softmax will equal to 1, but sigmoid not. Therefore, the sigmoid function is used for binary classification problems and in neural networks as an activation function, while the softmax function is used for multi-class classification problems and in neural networks as an output activation function.

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

Ans:

In classification tasks, the cross-entropy loss function is preferred because it is better able to penalize incorrect predictions of probabilities. Cross-entropy measures the dissimilarity between the predicted and actual probability distributions, where a larger dissimilarity results in a larger loss. This is important for classification tasks because we want to avoid assigning high probabilities to incorrect classes, which could lead to misclassifications.

In regression tasks, the mean squared error loss function is preferred because it penalizes large errors in the predictions. Mean squared error measures the average squared difference between the predicted and actual values, where a larger difference results in a larger loss. This is important for regression tasks because we want to minimize the differences between the predicted and actual values, as large differences could lead to inaccurate predictions.

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

Ans:

Multiply every data by a constant. It is used to enlarge the S_B.

(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.)

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

$$\frac{\partial}{\partial x}\left(\left(\mathrm{y} - \sigma(x)\right)^2\right)$$

$\sigma(x)$: sigmoid function $= \hat{y}$

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

$= \frac{\partial \hat{y}}{\partial x} \cdot \frac{\partial}{\partial \hat{y}}$

$= \frac{\partial \hat{y}}{\partial x} \cdot \left(y \cdot \frac{1}{\hat{y}} + (1-y) \cdot \frac{1}{1-\hat{y}} \cdot (-1)\right)$

$= x(1-x)\left(y \cdot \frac{1}{\sigma(x)} - (1-y)\frac{1}{1-\sigma(x)}\right)$

(2) $\frac{\partial}{\partial x}\left((y-\sigma(x))^2\right)$

$= \frac{\partial \hat{y}}{\partial x} \cdot \frac{\partial}{\partial \hat{y}}$

$= \frac{\partial \hat{y}}{\partial x} \cdot (-2y + 2\hat{y})$

$= x(1-x)(-2y + 2\sigma(x))$