STAT 542: Homework 2

Ahmadreza Eslaminia

Ae15

Please make sure that your solutions are readable and the file size is reasonable. Typing the answers is highly encouraged.

Problem 1.

[2pts] Complete Exercise 4.2 in ESL to show the equivalence of LDA and linear regression in a certain setting. Our Notes LDA.pdf on Canvas already provided most of the ingredients of the calculations. Address the following question to complete the exercise:

- Let $\hat{\delta}_1(x)$ and $\hat{\delta}_2(x)$ be the discriminant functions in LDA (note that the condition in part (a) of Ex 4.2 is $\hat{\delta}_2(x) > \hat{\delta}_1(x)$). Let $\hat{f}(x)$ be the linear regression function in part (e). Show that when $N_1 = N_2$ we have

$$\hat{f}(x) = \lambda \left(\hat{\delta}_2(x) - \hat{\delta}_1(x) \right) \tag{1}$$

for some scalar $\lambda > 0$ depending on the training data (but not on the new input x). What is λ ?

Solution

4.2.a

The LDA rule assigns x to the class for which the posterior probability is greatest. The inequality for classification to class 2 can be derived as follows:

$$\begin{gathered}
P(Y=2|X=x) > P(Y=1|X=x) \\
\frac{P(X=x|Y=2)P(Y=2)}{P(X=x)} > \frac{P(X=x|Y=1)P(Y=1)}{P(X=x)} \\
P(X=x|Y=2)P(Y=2) > P(X=x|Y=1)P(Y=1) \\
\exp\left(-\frac{1}{2}(x-\mu_2)^T\Sigma^{-1}(x-\mu_2)\right)\cdot\frac{N_2}{N} > \exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)\cdot\frac{N_1}{N}
\end{gathered}$$

By taking logarithms and rearranging, we obtain the LDA decision rule similar to equation 4.9 in the textbook which is equal to the one in this question:

$$x^T\Sigma^{-1}(\mu_2-\mu_1)-\frac{1}{2}(\mu_2+\mu_1)^T\Sigma^{-1}(\mu_2-\mu_1)+\log\left(\frac{N_2}{N_1}\right)>0 \tag{1}$$

Thus, according to the LDA rule, a feature vector x is classified to class 2 if the above inequality holds, and to class 1 otherwise.

4.2.b

Let's consider the feature vector for each observation i as $\mathbf{x}_i=(x_{i1},\ldots,x_{ip})^T\in\mathbb{R}^p$, and the design matrix $\mathbf{X}\in\mathbb{R}^{N\times(p+1)}$ which includes a column of ones to incorporate the intercept term in our model. The response vector is $\mathbf{y}\in\mathbb{R}^N$.

The design matrix $\mathbf{X}$ and the corresponding transpose $\mathbf{X}^T$ are given by:

$$\mathbf{X}=\begin{bmatrix}1 & x_{11} & \cdots & x_{1p}\\ 1 & x_{21} & \cdots & x_{2p}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & x_{N1} & \cdots & x_{Np}\end{bmatrix}=\begin{bmatrix}1 & \mathbf{x}_1^T\\ 1 & \mathbf{x}_2^T\\ \vdots & \vdots\\ 1 & \mathbf{x}_N^T\end{bmatrix},$$

$$\mathbf{X}^T=\begin{bmatrix}1 & 1 & \cdots & 1\\ x_{11} & x_{21} & \cdots & x_{N1}\\ \vdots & \vdots & \ddots & \vdots\\ x_{1p} & x_{2p} & \cdots & x_{Np}\end{bmatrix}=\begin{bmatrix}1 & 1 & \cdots & 1\\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N\end{bmatrix}.$$

Consequently, we have the following expression for the product of $\mathbf{X}^T$ and $\mathbf{X}$:

$$\mathbf{X}^T\mathbf{X}\begin{bmatrix}\beta_0\\ \boldsymbol{\beta}\end{bmatrix}=\begin{bmatrix}N & \sum_{i=1}^N\mathbf{x}_i^T\\ \sum_{i=1}^N\mathbf{x}_i & \sum_{i=1}^N\mathbf{x}_i\mathbf{x}_i^T\end{bmatrix}\begin{bmatrix}\beta_0\\ \boldsymbol{\beta}\end{bmatrix}=\mathbf{X}^T\mathbf{y} \tag{2}$$

Also we know:

$$\mathbf{X}^T\mathbf{Y}=\begin{bmatrix}\sum_{i=1}^N y_i\\ \sum_{i=1}^N y_i\mathbf{x}_i\end{bmatrix}$$

This yields:

$$\begin{aligned}N\beta_0+\left(\sum_{i=1}^N\mathbf{x}_i^T\right)\boldsymbol{\beta}&=\sum_{i=1}^N y_i\\ \beta_0\sum_{i=1}^N\mathbf{x}_i+\left(\sum_{i=1}^N\mathbf{x}_i\mathbf{x}_i^T\right)\boldsymbol{\beta}&=\sum_{i=1}^N y_i\mathbf{x}_i\end{aligned} \tag{3}$$

By isolating the term for the intercept β_0 from the first equation, we obtain:

$$\beta_0=\frac{1}{N}\left(\sum_{i=1}^N y_i-\left(\sum_{i=1}^N\mathbf{x}_i^T\right)\boldsymbol{\beta}\right), \tag{4}$$

By inserting the β_0 into equation (3) we can drive:

$$\left(\sum_{i=1}^{N} x_i x_i^T - \frac{1}{N}\left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} x_i^T\right)\right)\beta = \sum_{i=1}^{N} y_i x_i - \frac{1}{N}\left(\sum_{i=1}^{N} y_i\right)\left(\sum_{i=1}^{N} x_i\right) \tag{5}$$

At this juncture, we introduce the class-specific means, $\hat{\mu}_1$ and $\hat{\mu}_2$, and the class encoding for y_i as c_1 and c_2 for class 1 and class 2, respectively. With $c_1 = -N/N_1$ and $c_2 = N/N_2$, and by definition, the class means are:

$$\hat{\mu}_1 = \frac{1}{N_1}\sum_{i:g_i=1} \mathbf{x}_i,$$
$$\hat{\mu}_2 = \frac{1}{N_2}\sum_{i:g_i=2} \mathbf{x}_i.$$

We then express the sums involving $\mathbf{x}_i$ in terms of $\hat{\mu}_1$ and $\hat{\mu}_2$:

$$\sum_{i=1}^{N} \mathbf{x}_i = N_1\hat{\mu}_1 + N_2\hat{\mu}_2,$$

Similarly, we encode the sums involving y_i:

$$\sum_{i=1}^{N} y_i = N_1 c_1 + N_2 c_2,$$
$$\sum_{i=1}^{N} y_i \mathbf{x}_i = c_1 N_1 \hat{\mu}_1 + c_2 N_2 \hat{\mu}_2.$$

Given the pooled covariance matrix $\hat{\Sigma}$, we have:

$$\hat{\Sigma} = \frac{1}{N-2}\left(\sum_{i:g_i=1} (\mathbf{x}_i - \hat{\mu}_1)(\mathbf{x}_i - \hat{\mu}_1)^T + \sum_{i:g_i=2} (\mathbf{x}_i - \hat{\mu}_2)(\mathbf{x}_i - \hat{\mu}_2)^T\right). \tag{6}$$

From this definition, we can expand the sums and express $\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T$ in terms of the class means $\hat{\mu}_1$ and $\hat{\mu}_2$:

$$\begin{aligned}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T &= \sum_{i:g_i=1}\mathbf{x}_i\mathbf{x}_i^T + \sum_{i:g_i=2}\mathbf{x}_i\mathbf{x}_i^T \\ &= \sum_{i:g_i=1}(\mathbf{x}_i\mathbf{x}_i^T - \hat{\mu}_1\mathbf{x}_i^T + \hat{\mu}_1\mathbf{x}_i^T) + \sum_{i:g_i=2}(\mathbf{x}_i\mathbf{x}_i^T - \hat{\mu}_2\mathbf{x}_i^T + \hat{\mu}_2\mathbf{x}_i^T) \\ &= \sum_{i:g_i=1}(\mathbf{x}_i - \hat{\mu}_1)(\mathbf{x}_i - \hat{\mu}_1)^T + \sum_{i:g_i=2}(\mathbf{x}_i - \hat{\mu}_2)(\mathbf{x}_i - \hat{\mu}_2)^T + N_1\hat{\mu}_1\hat{\mu}_1^T + N_2\hat{\mu}_2\hat{\mu}_2^T \\ &= (N-2)\hat{\Sigma} + N_1\hat{\mu}_1\hat{\mu}_1^T + N_2\hat{\mu}_2\hat{\mu}_2^T.\end{aligned}$$

Now we consider the left-hand side (LHS) of equation (5):

$$\begin{aligned}\sum_{i=1}^{N}x_ix_i^T - \frac{1}{N}\left(\sum_{i=1}^{N}x_i\right)\left(\sum_{i=1}^{N}x_i^T\right) &= \sum_{i=1}^{N}x_ix_i^T - \frac{1}{N}(N_1\hat{\mu}_1 + N_2\hat{\mu}_2)(N_1\hat{\mu}_1^T + N_2\hat{\mu}_2^T) \\ &= (N-2)\hat{\Sigma} + \frac{N_1N_2}{N}(\hat{\mu}_2 - \hat{\mu}_1)(\hat{\mu}_2 - \hat{\mu}_1)^T \\ &= (N-2)\hat{\Sigma} + N\hat{\Sigma}_B, \qquad (7)\end{aligned}$$

where $\hat{\Sigma}_B = \frac{N_1N_2}{N^2}(\hat{\mu}_2 - \hat{\mu}_1)(\hat{\mu}_2 - \hat{\mu}_1)^T$.

On the right-hand side (RHS) of equation (5), we have:

$$\begin{aligned}\sum_{i=1}^{N}y_ix_i - \frac{1}{N}\left(\sum_{i=1}^{N}y_i\right)\left(\sum_{i=1}^{N}x_i\right) &= c_1N_1\hat{\mu}_1 + c_2N_2\hat{\mu}_2 - \frac{1}{N}(N_1c_1 + N_2c_2)(N_1\hat{\mu}_1 + N_2\hat{\mu}_2) \\ &= \frac{N_1N_2}{N}(c_2 - c_1)(\hat{\mu}_2 - \hat{\mu}_1), \qquad (8)\end{aligned}$$

Combining previous equations we arrive at:

$$[(N-2)\hat{\Sigma} + N\hat{\Sigma}_B]\hat{\beta} = \frac{N_1N_2}{N}(c_2 - c_1)(\hat{\mu}_2 - \hat{\mu}_1). \qquad (9)$$

noting that $c_1 = -N/N_1$, $c_2 = N/N_2$ and thus $c_2 - c_1 = \frac{N^2}{N_1N_2}$ which means:

$$\left[(N-2)\hat{\Sigma} + N\hat{\Sigma}_B\right]\hat{\beta} = N(\hat{\mu}_2 - \hat{\mu}_1)$$

4.2.c

We aim to demonstrate that the least-squares regression coefficient vector $\hat{\beta}$ is proportional to the difference in class means, $\hat{\mu}_2 - \hat{\mu}_1$, effectively showing that $\hat{\beta}$ is aligned with the LDA discriminant vector. From the previous sections, we have:

$$\hat{\Sigma}_B\hat{\beta} = \frac{N_1N_2}{N^2}(\hat{\mu}_2 - \hat{\mu}_1)(\hat{\mu}_2 - \hat{\mu}_1)^T\hat{\beta}$$

Given the scalar nature of $(\hat{\mu}_2 - \hat{\mu}_1)^T\hat{\beta}$, we can see that $\hat{\Sigma}_B\hat{\beta}$ is in the direction of $(\hat{\mu}_2 - \hat{\mu}_1)$. Therefore:

$$\hat{\beta} \propto \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1)$$

4.2.d

To show that the relationship between the least-squares regression coefficients and the LDA coefficients holds for any coding of the two classes, we consider the equation:

$$[(N-2)\hat{\Sigma} + N\lambda\hat{\Sigma}_B]\hat{\beta} = \frac{N_1N_2}{N}(c_2 - c_1)(\hat{\mu}_2 - \hat{\mu}_1) \tag{10}$$

where $\hat{\Sigma}_B$ is defined as the between-class scatter matrix:

$$\hat{\Sigma}_B = \frac{N_1N_2}{N^2}(\hat{\mu}_2 - \hat{\mu}_1)(\hat{\mu}_2 - \hat{\mu}_1)^T \tag{11}$$

By multiplying out the left-hand side of equation (1) and using the definition of $\hat{\Sigma}_B$ from equation (2), we can see that $\hat{\Sigma}_B\hat{\beta}$ is indeed in the direction of $(\hat{\mu}_2 - \hat{\mu}_1)$, as the term $(\hat{\mu}_2 - \hat{\mu}_1)^T\hat{\beta}$ results in a scalar. Hence, the product $\hat{\Sigma}_B\hat{\beta}$ is a scalar multiple of the vector $(\hat{\mu}_2 - \hat{\mu}_1)$.

The scalar multiple itself is $\frac{N_1N_2}{N}(c_2 - c_1)$ divided by the sum of $(N-2)$ and $N\lambda$, which are coefficients that do not change the direction of $\hat{\beta}$. Therefore, regardless of the values assigned to c_1 and c_2, the directionality of $\hat{\beta}$ is preserved. This leads us to the conclusion that:

$$\hat{\beta} \propto \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) \tag{12}$$

This confirms that the least-squares regression coefficient vector is directionally equivalent to the LDA coefficient vector for any coding of the classes, differing only by a scalar multiple.

4.2.e

We proceed to find the solution for the intercept $\hat{\beta}_0$ and the corresponding predicted value function $\hat{f}(x)$ based on the provided encoding scheme.

Starting with the encoding of $-N/N_1$ and N/N_2 for classes 1 and 2 respectively, we have the intercept as:

$$\hat{\beta}_0 = \frac{1}{N}\left(\sum_{i=1}^{N} y_i - \left(\sum_{i=1}^{N} x_i^T\right)\beta\right), \tag{13}$$

when $N_1 = N_2$ simplifies to:

$$\hat{\beta}_0 = -\frac{1}{N}\left(N_1\hat{\mu}_1^T + N_2\hat{\mu}_2^T\right)\beta. \tag{14}$$

Thus, the predicted value function is:

$$\hat{f}(x) = \hat{\beta}_0 + x^T\beta = x^T\beta - \frac{1}{N}\left(N_1\hat{\mu}_1^T + N_2\hat{\mu}_2^T\right)\beta. \tag{15}$$

Since we have previously established that $\beta \propto \Sigma^{-1}(\hat{\mu}_2 - \hat{\mu}_1)$, there exists a scalar $\lambda > 0$ such that:

$$\beta = \lambda\Sigma^{-1}(\hat{\mu}_2 - \hat{\mu}_1). \tag{16}$$

Therefore, the classification rule derived from the sign of $\hat{f}(x)$ is consistent with the LDA rule when $N_1 = N_2$. However, when $N_1 \neq N_2$, the log term in the LDA rule introduces an adjustment that is not present in the regression approach, leading to a difference in the decision boundary unless the classes have equal numbers of observations.

4.2.f

showing that when the number of observations in both classes is equal, i.e., $N_1 = N_2$, the linear regression function $\hat{f}(x)$ is proportional to the difference in discriminant functions from LDA.

Recall the discriminant functions from LDA for two classes:

$$\hat{\delta}_1(x) = x^T\Sigma^{-1}\hat{\mu}_1 - \frac{1}{2}\hat{\mu}_1^T\Sigma^{-1}\hat{\mu}_1 + \log P(Y = 1),$$
$$\hat{\delta}_2(x) = x^T\Sigma^{-1}\hat{\mu}_2 - \frac{1}{2}\hat{\mu}_2^T\Sigma^{-1}\hat{\mu}_2 + \log P(Y = 2).$$

The linear regression function from part (e) is given by:

$$\hat{f}(x) = \hat{\beta}_0 + x^T\hat{\beta}, \tag{17}$$

where $\hat{\beta}_0$ is the intercept and $\hat{\beta}$ is the vector of coefficients.

Given that $\hat{\beta}$ is proportional to $\Sigma^{-1}(\hat{\mu}_2 - \hat{\mu}_1)$, we can write:

$$\hat{\beta} = \lambda\Sigma^{-1}(\hat{\mu}_2 - \hat{\mu}_1) \tag{18}$$

for some scalar $\lambda > 0$.

The intercept $\hat{\beta}_0$ can be expressed as:

$$\hat{\beta}_0 = -\frac{1}{N}(N_1\hat{\mu}_1^T + N_2\hat{\mu}_2^T)\hat{\beta}. \tag{19}$$

Substituting the expressions for $\hat{\beta}_0$ and $\hat{\beta}$ into $\hat{f}(x)$, and noting that when $N_1 = N_2$, the log prior terms cancel each other out, we get:

$$\hat{f}(x) = \lambda\left(x^T\Sigma^{-1}(\hat{\mu}_2 - \hat{\mu}_1) - \frac{1}{2}(\hat{\mu}_1^T\Sigma^{-1}\hat{\mu}_1 - \hat{\mu}_2^T\Sigma^{-1}\hat{\mu}_2)\right). \tag{20}$$

This is equivalent to $\lambda(\hat{\delta}_2(x) - \hat{\delta}_1(x))$ when $N_1 = N_2$. Therefore, $\hat{f}(x)$ is a scalar multiple of the LDA discriminant difference, with λ being the scalar that adjusts for the difference in magnitudes of the coefficients.

Problem 2.

[2pts] Run the iris dataset example for lda documentation: https://www.rdocumentation.org/packages/MASS/v 58.2/topics/lda After training the model, write codes to use the test set to compute the confusion matrix and then the values of sensitivity and specificity (see s6_logistic.pdf for definitions). Please include a screenshot to show your codes and results after running the code.

Hint: In the lecture we defined sensitivity and specificity for the case of binary class, but in the multiclass case as in the homework, we can use a "one-against-all" approach to extend the definition; see how-to-calculate-multiclass-overall-accuracy-sensitivity-and-specificity

Solution2

The code is in Python and attached below:

C: > DriveA > UIUCcourses > Spring 2024 > Stat 542 Statistical Learning > Hws > 2 > Q2.py > ...

```python
from sklearn import datasets
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Initialize LDA and fit it to the training data
lda = LDA()
lda.fit(X_train, y_train)

# Predict the test set results
y_pred = lda.predict(X_test)

# Compute the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Print the confusion matrix
print("Confusion Matrix:")
print(cm)

# Calculate sensitivity and specificity
report = classification_report(y_test, y_pred, output_dict=True)

# Extracting sensitivity and specificity for each class
sensitivity = {}
specificity = {}

for label, metrics in report.items():
    if label.isdigit():  # Check if the key is a digit to ignore 'accuracy', 'macro avg', etc.
        sensitivity[label] = metrics['recall']
```

Figure 1: Code

results:

```python
# Extracting sensitivity and specificity for each class
sensitivity = {}
specificity = {}

for label, metrics in report.items():
    if label.isdigit():  # Check if the key is a digit to ignore 'accuracy', 'macro avg', etc.
        sensitivity[label] = metrics['recall']

        # Specificity is the true negative rate
        true_negatives = cm.sum() - (cm[int(label), :].sum() + cm[:, int(label)].sum() - cm[int(label), int(label)])
        specificity[label] = true_negatives / (true_negatives + cm[:, int(label)].sum() - cm[int(label), int(label)])

# Print sensitivity and specificity for each class
print("\nSensitivity (Recall) for each class:")
for label, value in sensitivity.items():
    print(f"Class {label}: {value}")

print("\nSpecificity for each class:")
for label, value in specificity.items():
    print(f"Class {label}: {value}")
```

Figure 2: code continue

```
45  for label, value in sensitivity.items():
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
ncher' '60674' '--' 'C:\DriveA\UIUCcourses\Spring 2024\Stat 542 Statistical Learning\Hws\2\Q2.py'
Confusion Matrix:
[[16  0  0]
 [ 0 17  1]
 [ 0  0 11]]

Sensitivity (Recall) for each class:
Class 0: 1.0
Class 1: 0.9444444444444444
Class 2: 1.0

Specificity for each class:
Class 0: 1.0
Class 1: 1.0
Class 2: 0.9705882352941176
PS C:\DriveA\UIUCcourses\Spring 2024\Stat 542 Statistical Learning\Hws\2>
```

Figure 3: code result

Problem 3.

[3pts] Repeat Problem 2 using logistic regression instead of LDA. Compare the errors (sensitivity and specificity) with the LDA method. Also compare the performances with and without the intercept.

Solution3

The code is in Python and attached below:

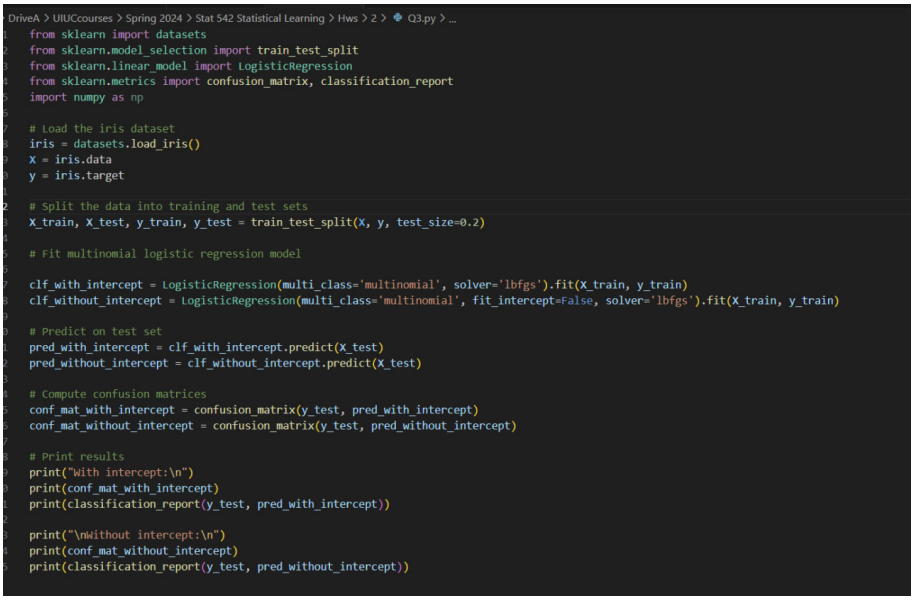

Figure 4: Code

results:

The model's performance improves when including the intercept term, which is expected because the intercept allows the decision boundary to shift away from the origin. This flexibility is essential in real-world scenarios where the boundary is not anticipated to pass through the origin, especially when the data are not normalized or centered.

In comparing LDA with logistic regression, we find that LDA outperforms logistic regression in terms of overall performance metrics. Specifically, LDA demonstrates comparable or superior performance to logistic regression with the intercept term included, and both LDA and logistic regression with the intercept term outperform logistic regression without the intercept term. Therefore, we can conclude that LDA is generally better than logistic regression for this particular task.

```
With intercept:

[[ 8  0  0]
 [ 0 10  1]
 [ 0  0 11]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      0.91      0.95        11
           2       0.92      1.00      0.96        11

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```

Figure 5: code results

```
Without intercept:

[[ 8  0  0]
 [ 0 10  2]
 [ 0  0 10]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      0.83      0.91        12
           2       0.83      1.00      0.91        10

    accuracy                           0.93        30
   macro avg       0.94      0.94      0.94        30
weighted avg       0.94      0.93      0.93        30
```

Figure 6: code result

Problem 4.

[1 bonus point] In our slides page "Advantages of LDA over logistic regression", it was mentioned that logistic regression may be unstable in certain settings. Consider the following example: In binary logistic regression, suppose that $p > n, X$ has full row rank(rank $= n$), and there is no intercept. Show that the likelihood function is not maximized by any finite β; in fact we can let β diverge while the likelihood tends to 1 .

Hint: pick a β so that there is no training error in classification, and then set $\beta \leftarrow t\beta$ with the scalar $t \to \infty$.

Solution 4

The statement about the instability of the maximum likelihood estimate (MLE) in logistic regression when $p > n$ and there is perfect separation can be shown mathematically. Perfect separation means that there exists some parameter vector β such that the logistic function perfectly separates the two classes.

Given the log likelihood for logistic regression:

$$\ell(\beta) = \sum_{i=1}^{n} \log\left(\frac{1}{1 + \exp(-y_i x_i^T \beta)}\right)$$

Now, consider the case of perfect separation. This means for each i, $y_i x_i^T \beta$ is positive and large for $y_i = 1$ and negative and large for $y_i = -1$. Therefore, $\exp(-y_i x_i^T \beta)$ will be close to zero for all i.

As $t \to \infty$, let $\beta_t = t\beta$, where β is the vector that perfectly separates the data. Then:

$$\lim_{t\to\infty} \exp(-y_i x_i^T \beta_t) = \lim_{t\to\infty} \exp(-t y_i x_i^T \beta) = 0$$

Since the exponential term approaches zero, the log likelihood becomes:

$$\ell(\beta_t) = \sum_{i=1}^{n} \log\left(\frac{1}{1+0}\right) = \sum_{i=1}^{n} \log(1) = 0$$

However, this is an issue because as t increases, the β parameters become increasingly large, and the likelihood does not approach a finite maximum but instead goes to zero. This makes the MLE not well-defined, as there's no unique solution that maximizes the likelihood—the likelihood can be increased indefinitely by increasing the magnitude of β. In practical terms, this can lead to overfitting, where the model predictions are too closely tied to the particularities of the sample and may not generalize well to new data. This situation is an example of what is called the "Hauck-Donner phenomenon," where the likelihood approaches its supremum, but no maximum likelihood estimate exists because it would require β to go to infinity.