Linear Discriminant Analysis: From Scratch Implementation to Classification

Aman Vaibhav Jha
2201AI54

September 19, 2025

Abstract

This report details the implementation of the Linear Discriminant Analysis (LDA) algorithm from scratch in Python. The project demonstrates the dual utility of LDA: first as a tool for dimensionality reduction to visualize high-dimensional data, and second as a preprocessing step for a supervised classification task. The algorithm was applied to the Iris dataset, successfully projecting its four features into a two-dimensional space that maximizes class separability. A Logistic Regression classifier trained on this reduced data achieved excellent performance, which was evaluated using a confusion matrix, classification report, and visual analysis of test set predictions.

Contents

1 Introduction

Linear Discriminant Analysis (LDA) is a powerful supervised learning algorithm used for both dimensionality reduction and classification. Unlike unsupervised techniques like PCA which maximize variance, LDA's objective is to find a feature subspace that maximizes the separability between classes. This makes it particularly effective as a preprocessing step for classification models.

This project covers the end-to-end process of:

- Implementing the LDA algorithm from its mathematical foundations using NumPy.
- Applying LDA to the classic Iris dataset for exploratory visualization.
- Evaluating the performance of a classifier trained on the LDA-transformed data.

2 Methodology and Theory

2.1 The Mathematical Foundation of LDA

The primary goal of LDA is to find a transformation matrix $\mathbf{W}$ that projects the data from a d-dimensional space to a d'-dimensional space (where $d' < d$) by maximizing the ratio of between-class variance to within-class variance. This objective is often expressed by Fisher's criterion:

$$J(\mathbf{W}) = \frac{|\mathbf{W}^T S_B \mathbf{W}|}{|\mathbf{W}^T S_W \mathbf{W}|}$$

where S_B is the between-class scatter matrix and S_W is the within-class scatter matrix. The process to find the optimal $\mathbf{W}$ involves the following steps:

1. Compute Mean Vectors For a dataset with k classes, we first compute the mean vector for each class C_i and the overall mean vector. Let N_i be the number of samples in class i.

- **Class Mean Vector ($\boldsymbol{\mu}_i$):** The mean of all samples belonging to class i.

$$\boldsymbol{\mu}_i = \frac{1}{N_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

- **Overall Mean Vector ($\boldsymbol{\mu}$):** The mean of all samples in the dataset.

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{x}_j = \frac{1}{N} \sum_{i=1}^{k} N_i \boldsymbol{\mu}_i$$

2. Compute the Scatter Matrices The scatter matrices quantify the variance within and between classes.

- **Within-Class Scatter Matrix (S_W):** This matrix measures the spread of samples around their respective class means. It is the sum of the scatter matrices for each class. A small S_W indicates that samples within each class are tightly clustered.

$$S_W = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T$$

- **Between-Class Scatter Matrix (S_B):** This matrix measures the spread of the class means around the overall dataset mean. A large S_B indicates that the class means are far apart from each other.

$$S_B = \sum_{i=1}^{k} N_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$$

3. Solve the Generalized Eigenvalue Problem Maximizing the objective function $J(\mathbf{W})$ is equivalent to solving the generalized eigenvalue problem for the matrix $S_W^{-1} S_B$. The problem is stated as:

$$S_B \mathbf{v} = \lambda S_W \mathbf{v}$$

Assuming S_W is non-singular (invertible), we can rewrite this as a standard eigenvalue problem:

$$S_W^{-1} S_B \mathbf{v} = \lambda \mathbf{v}$$

The solutions to this problem are:

- The **eigenvectors (v)** represent the directions of the new feature space, known as the linear discriminants. These are the directions that maximize class separation.
- The **eigenvalues (λ)** represent the ratio of between-class scatter to within-class scatter for each corresponding eigenvector. Larger eigenvalues correspond to more discriminative power.

4. Construct the Transformation Matrix We sort the eigenvectors in descending order based on their corresponding eigenvalues. We then select the top d' eigenvectors to create the transformation matrix $\mathbf{W}$, where each column is an eigenvector:

$$\mathbf{W} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{d'}]$$

For a problem with k classes, the maximum number of useful discriminants is $k - 1$.

5. Project the Data Finally, the original d-dimensional data $\mathbf{X}$ is transformed into the new d'-dimensional space $\mathbf{Y}$ using the matrix $\mathbf{W}$:

$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{W}$$

2.2 Classification Pipeline

To evaluate the effectiveness of LDA, a Logistic Regression classifier was trained. The projected data was split into an 80% training set and a 20% test set. The model was trained only on the training data and evaluated on the unseen test data.

3 Results and Analysis

The implementation was performed in a Jupyter Notebook. The following sections present the key outputs.

3.1 Part 1: Exploratory Visualization

LDA was first applied to the entire Iris dataset to visualize its dimensionality reduction capability. The 4-dimensional data was projected onto the top 2 linear discriminants. The resulting scatter plot is shown in Figure 1.

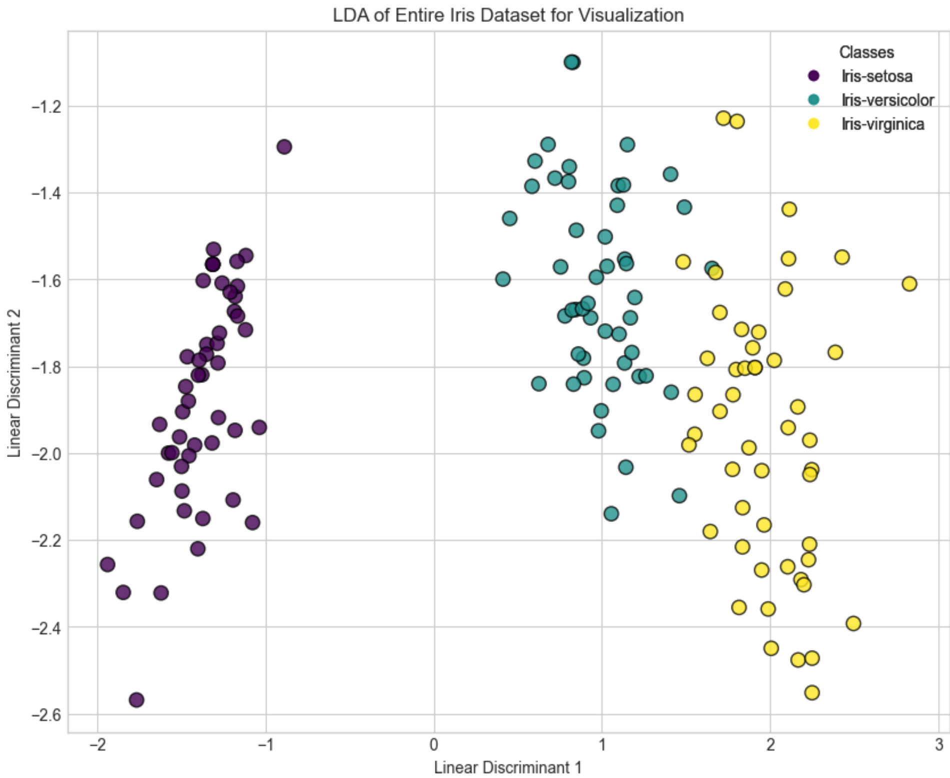

Figure 1: Visualization of the entire Iris dataset projected into 2D using LDA. The classes show excellent linear separability.

3.2 Part 2: Classification Performance

The classifier was evaluated on the 20% test set. The results are summarized below.

3.2.1 Confusion Matrix

The confusion matrix in Figure 2 shows the number of correct and incorrect predictions for each class. The strong diagonal indicates a high number of correct predictions.

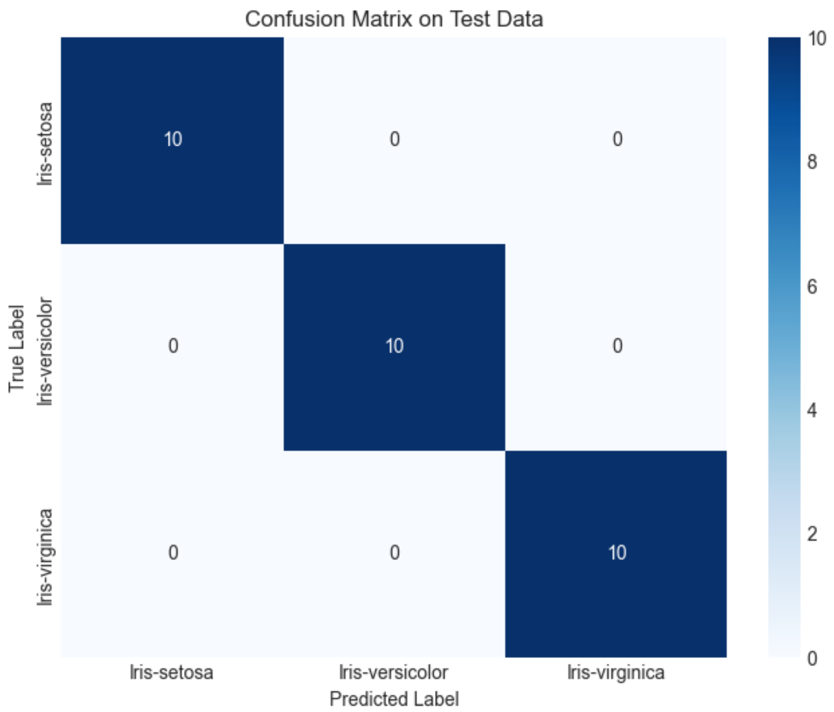

Figure 2: Confusion Matrix for the Logistic Regression classifier on the test set.

3.2.2 Classification Report

The classification report provides detailed metrics like precision, recall, and F1-score. The overall accuracy of the model was excellent.

```
                precision    recall  f1-score   support

 Iris-setosa         1.00      1.00      1.00        10
 Iris-versicolor     1.00      1.00      1.00        10
 Iris-virginica      1.00      1.00      1.00        11

    accuracy                             1.00        30
   macro avg         1.00      1.00      1.00        30
weighted avg         1.00      1.00      1.00        30
```

3.2.3 Test Set Visualization

Figure 3 visualizes the predictions on the test set. Misclassified points, if any, are circled in red. In this case, the model achieved perfect accuracy on the test set.

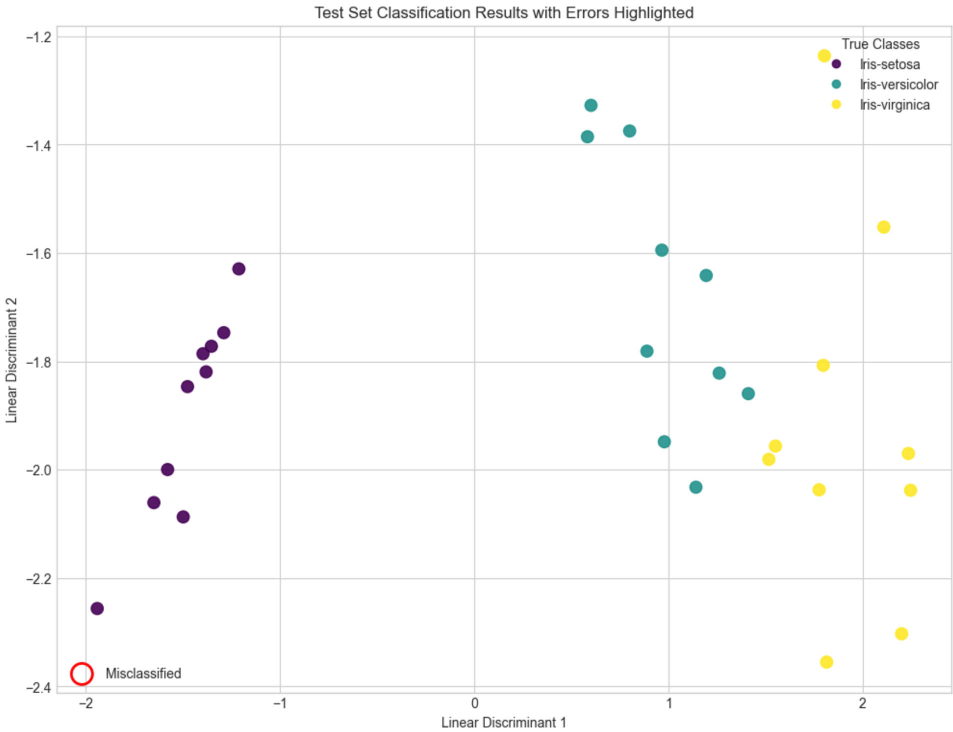

Figure 3: Classification results on the test set. All points were classified correctly.

4 Conclusion

This project successfully demonstrated the implementation and application of Linear Discriminant Analysis from scratch. The results confirm that LDA is a highly effective technique for both visualizing class separability and for preprocessing data for classification tasks. A simple linear classifier achieved perfect accuracy on the transformed Iris dataset, validating the power of the features extracted by LDA.