Coding Project 4: Teaching a Computer to Recognize Written Numbers

Jack

Abstract

In this project, we developed a machine learning algorithm to recognize handwritten digits. We utilized Linear Discriminant Analysis (LDA) as our classification method, and implemented a one-vs-all strategy for multi-class classification along with a k-NN model for comparison. After training the model with a dataset of handwritten digits, we evaluated its performance using a test dataset. The classifier demonstrated relatively poor accuracy in recognizing the handwritten digits, which highlights the ineffectiveness of LDA on non-Gaussian distributions.

1 Introduction

Machine learning is a powerful and rapidly evolving field that enables computers to learn from data and make predictions or decisions without being explicitly programmed. At its core, machine learning aims to develop algorithms that can identify patterns and relationships within data, allowing computers to "learn" from past experiences and improve their performance over time.

This report will provide an overview of our project, where we implemented a machine learning algorithm to recognize handwritten digits using Linear Discriminant Analysis (LDA). We will discuss the theoretical background of LDA, describe the implementation process, and present the results of our classifier's performance on the MNIST dataset purely classifying 1s and 0s. After this we attempt to run the same data through a K-nearest neighbors classifier to compare the results, and also check for all digits 0-9.

2 Theoretical Background

In this section, we will delve into the underlying theory and concepts related to our chosen classification method, Linear Discriminant Analysis, along with some brief introductory theory of K-nearest Neighbors.

2.1 Linear Discriminant Analysis vs K-nearest Neighbors

Linear Discriminant Analysis (LDA) is a method that assumes the data within each class follows a specific pattern and tries to find a linear transformation of the data that best separates the classes. LDA explicitly learns the patterns for each class, in our case here this would correspond to each unique digit. The method finds the optimal projection by solving the generalized eigenvalue problem for the between-class scatter matrix (S_B) and the within-class scatter matrix (S_W):

- $S_W = \sum_i (X - \mu_i)(X - \mu_i)^T$ (for each class i)
- $S_B = \sum_i N_i(\mu_i - \mu)(\mu_i - \mu)^T$ (for each class i)

where X is the data, μ_i is the mean of class i, N_i is the number of samples in class i, and μ is the overall mean. The eigenvectors corresponding to the largest eigenvalues are the linear discriminant components used to project the data into a lower-dimensional space while maximizing the class separation.Once the model is trained, LDA can quickly make predictions by calculating the class probabilities for the new data point. LDA's strengths include its fast prediction speed and its ability to provide an easy-to-understand, lower-dimensional representation of the data. However, LDA assumes that the data follows a Gaussian distribution within each class and that the classes have equal covariance matrices. If these assumptions are not true, LDA might not work as well.

K-nearest neighbors (k-NN) is significantly different. While it is a simple method that doesn't require building a model beforehand, it uses the entire training dataset when making predictions. Say we want to classify a new data point, k-NN looks at the k closest data points in the training set and assigns the most common class among those neighbors to the new data point. The main advantage of k-NN is its simplicity, and it can work well for many classification problems, especially when the boundaries between classes are

complicated. However, k-NN can be slow when making predictions, especially if the dataset is large, since it needs to calculate the distances to all training samples. Also, k-NN's performance can be sensitive to the choice of distance metric and the value of k, which may need fine-tuning for best results.

3 Results

Our Figure 1 is relatively simple, it's just an organization of singular values and showing how after around 15 or so singular values are used we capture significanlty less features in our data. This is why we ended up choosing 15 singular values.

In our second figure however we begin to see the large amount of overlap in both our classes. We should be attempting to minimize the total overlap which would indicate that the LDA projection has maximized the separation between them, which would result in a better classification performance. However we witness the exact opposite, which was incredibly worrying before attempting to create a one vs all model for the MNIST digits. This is when we decided to try k-NN and see a comparison. From our One v All model we were able to correctly classify 46% of our 5000 test digits, and it correctly guessed 7 of my 20 handwritten numbers (10 with my dominant hand, 10 with my weak hand). The k-NN did remarkably better however, where it correctly classified over 98% of our 5000 test digits and classified 17 of my hand digits properly (to be fair I think that would be better than most humans, my handwriting is abysmal not to mention somehow even worse with my weak hand). We can largely attribute the failure of the LDA model to the overlap in the LDA space from the classes, there distribution of the numbers simply was unable to confrom well to an LDA model.

4 Conclusion

In this project we utilized LDA and k-NN to attempt and create a multi-classifier for the MNIST data set. From this we have seen that k-NN's accuracy of 98% on all 10 digits is significantly better than the 46% accuracy LDA was able to produce. This is largely due to the non-Gaussian distribution why we deal with more than 2 digits, and k-NN excels immensely when doing this simple multi-classifications of similar inputs.

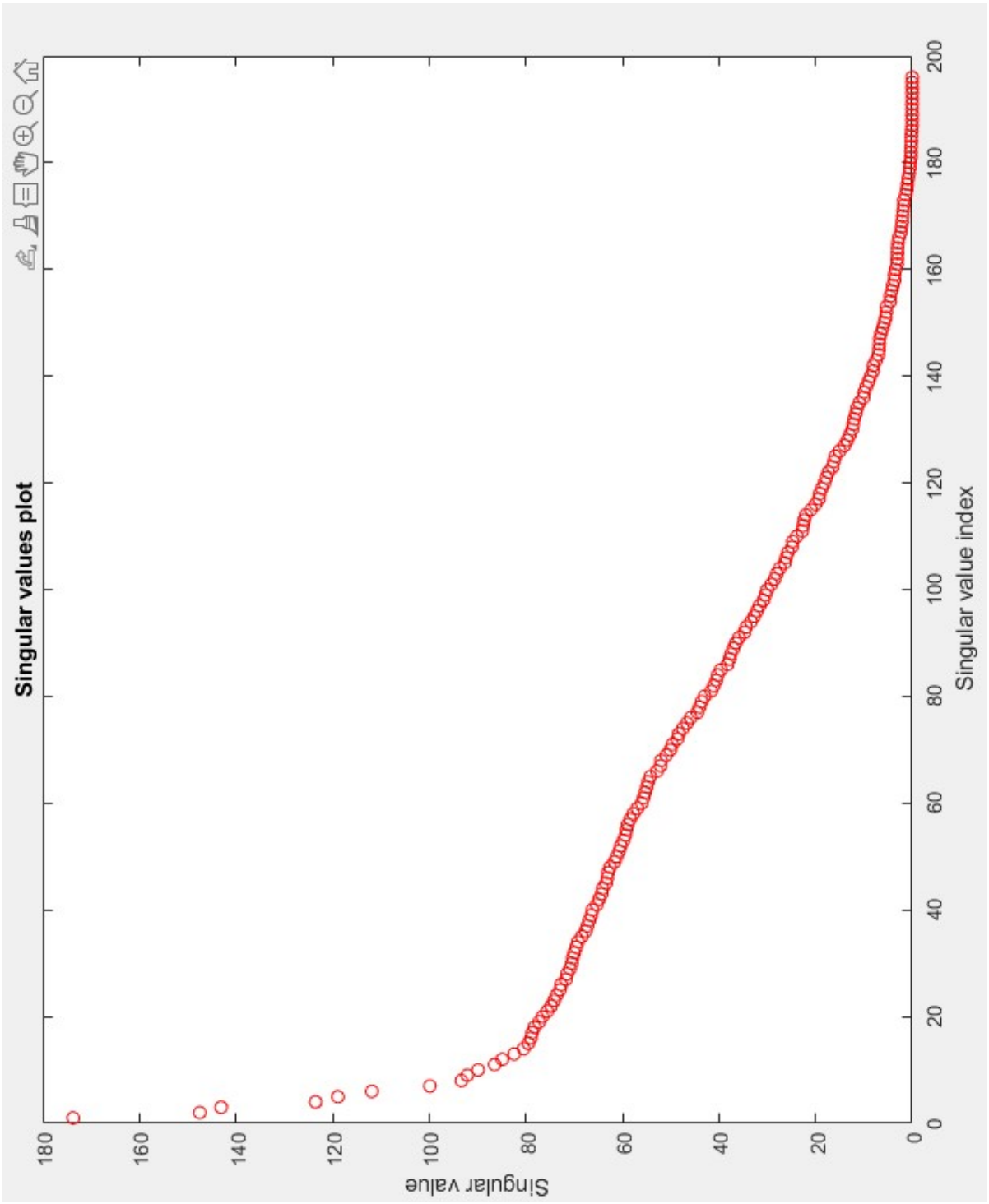

Figure 1: Singular Values

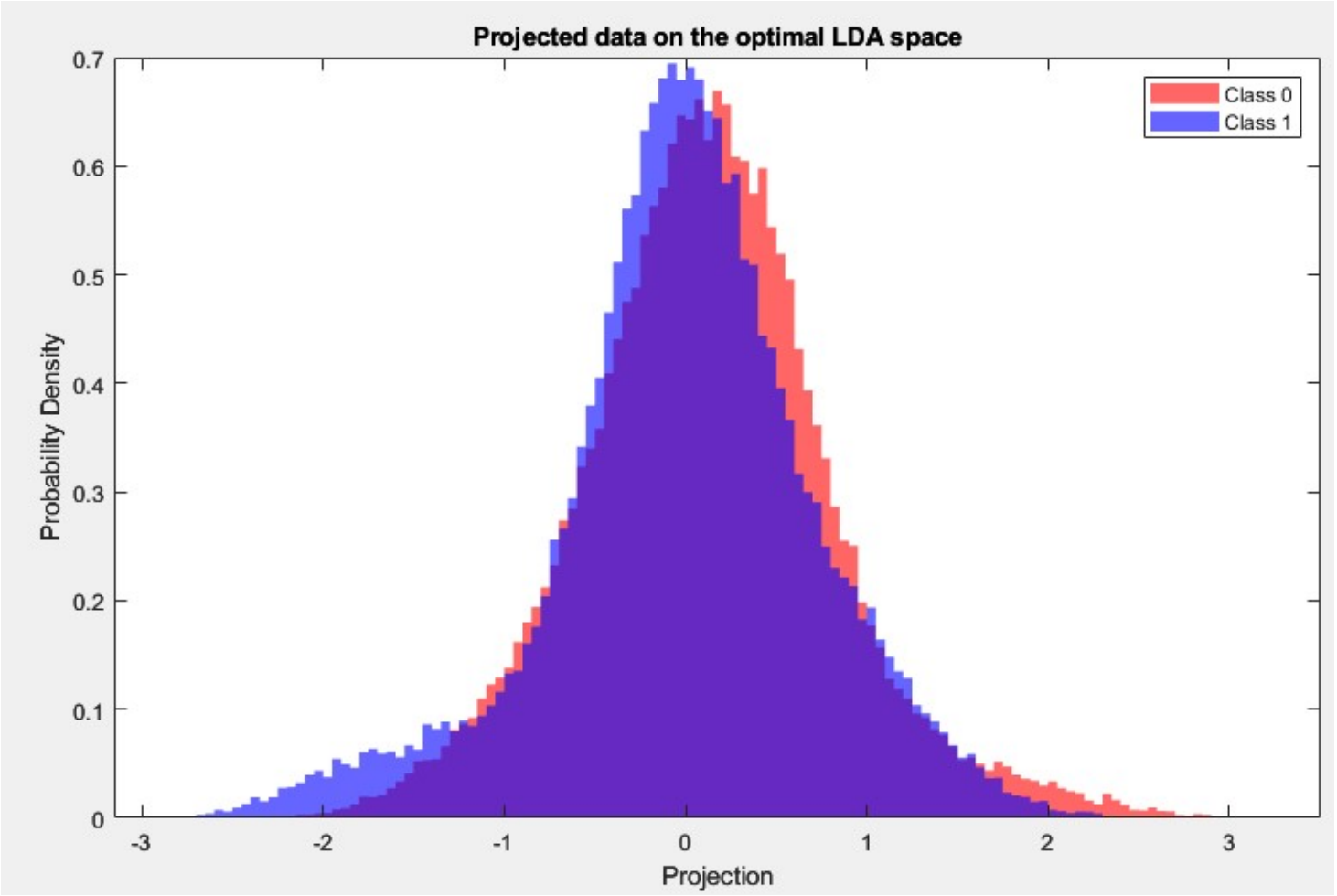

Figure 2: Projected Data onto LDA Space